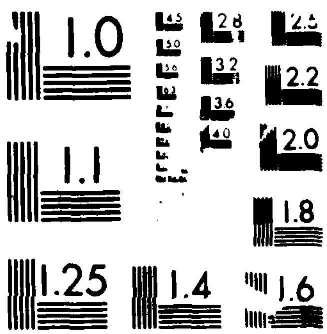
AD-A187 834 FEASIBILITY STUDY OF GENERATING PLANS AND STRATEGIES 1/2
FOR SOFTWARE TESTING..(U) SONEX ENTERPRISES INC FAIRFAX
VA D J MATCHINSKI 13 OCT 87 DAAB07-87-C-F012
UNCLASSIFIED F/G 12/5 NL



MICROCOPY RESOLUTION TEST CHART

(2)

Report DAAB07-87-C-F012

FEASIBILITY STUDY OF GENERATING PLANS AND STRATEGIES FOR SOFTWARE TESTING BY KNOWLEDGE BASED SYSTEM

Prepared for:
Center for Night Vision and Electro-Optics
ATTN: AMSEL-NV-AV (Tim Williams)
Fort Belvoir, VA 22060-5677

Commander US Army, CECOM
Product Assurance and Test Directorate
ATTN: AMSEL-PA-MT-S (Paul Kogut)
Fort Monmouth, NJ 07703

October 13, 1987

Final Report

SONEX ENTERPRISES INCORPORATED

By:
Donald J. Matchinski
Sonex Enterprises, Inc.
3998 Fair Ridge Drive
Suite 220
Fairfax, VA 22033
(703) 359-7033

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Feasibility Study for Generating Software Testing Plans and Strategies by Knowledge Based System		5. TYPE OF REPORT & PERIOD COVERED Final Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Donald J. Matchinski		8. CONTRACT OR GRANT NUMBER(s) DAAB07-87-C-F012
9. PERFORMING ORGANIZATION NAME AND ADDRESS Sonex Enterprises, Incorporated, 3998 Fair Ridge Drive, Suite 220, Fairfax, VA 22033		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Small Business Innovative Research
11. CONTROLLING OFFICE NAME AND ADDRESS Center for Night Vision & Electro-Optics, ATTN: AMSEL-NV-AV, Fort Belvoir, VA 22060-5677		12. REPORT DATE 13 Oct 87
		13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Commander, US CECOM, Product Assurance & Test Directorate, Fort Monmouth, NJ 07703, ATTN: AMSEL-PA-MT-S		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software testing, Knowledge based systems, testing architecture, DoD-STD-2167

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report documents a software testing architecture that encompasses the entire automated system life cycle. The architecture is designed with a knowledge based component that uses captured expert testing experience. The output of the system is a software test plan in accordance with DoD-STD-2167.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

FEASIBILITY STUDY FOR GENERATING SOFTWARE TESTING PLANS AND STRATEGIES BY KNOWLEDGE BASED SYSTEM

TABLE OF CONTENTS

LIST OF FIGURES

1.0 INTRODUCTION

The findings of Sonex Enterprises Incorporated under the Small Business Innovation Research (SBIR) Phase I program are contained within this report. The research was sponsored by the US Army, Communications-Electronics Command, Product Assurance and Test Directorate, Fort Monmouth, New Jersey.

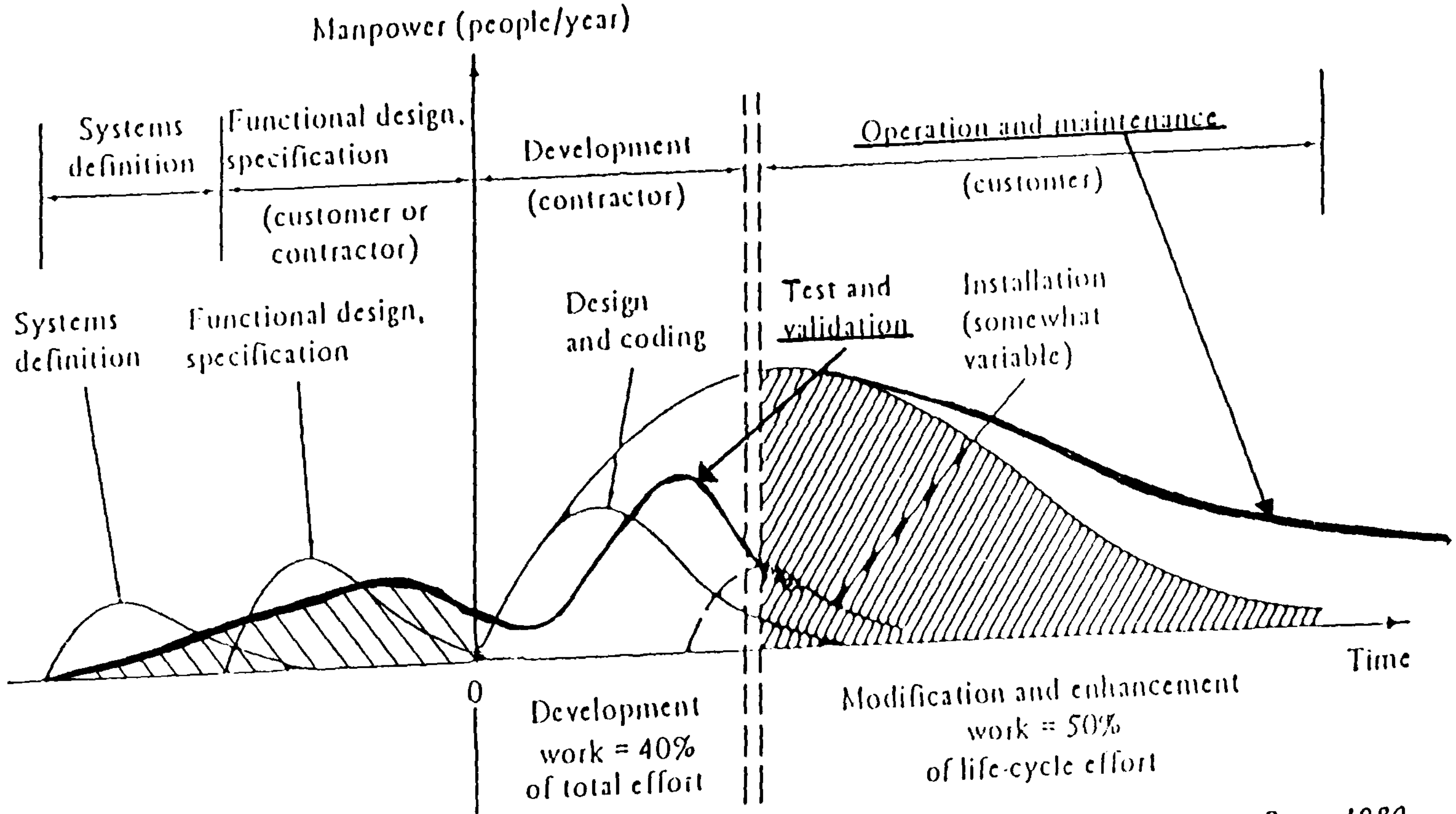
1.1 Problem Statement

The problem statement for this research was simplified to: Is it feasible today to produce a sound automated software test plan to support DoD system development and test activities? If it is feasible, how would the testing architecture be structured?

1.2 Scope of the Study

The current procurement/productivity environment demands a reduction in the system life cycle cost. The promise of fourth generation languages, relational data base technology, object oriented programming, re-useable software and other factors have led to systems with expensive and extended life cycles. The procurement environment requires that the potential run-away life cycle costs be constrained by establishing control at system inception. Testing is the only common control available for manager, programmer, contractor and customer. As the life cycle of automated system continues to be extended, the ability to test systems at all phases of development is accentuated. At Figure 1 is the graphic depicting the traditional Effort Distribution for Large Projects (Putnam, 1980). We have modified this figure to show the need for an increase and earlier start of the testing effort within the system life cycle. Our hypothesis is that additional testing must be included in the system definition and functional design specification phases to test the requirements before any code is written. Testing starts when the system definition starts -- because quality can not be incorporated into the system at the test phase near end of development, quality must be built into the system. The emphasis of this research has been to verify that true testability could be included in the system definition phase and functional design phases of system development.

Effort Distribution—Large Projects (Revised)



Source: L. Putnam, *Software Cost Estimating and Life Cycle Control, IEEE Computer Society Press, 1980.*

Figure 1

Figure 2 depicts the DoD defined levels of testing and the requirements documents/specifications consistant with MIL-STD-490 series documentation definitions. The focus of this research is the Acceptance level testing of the B5 specification.

1.3 Organization of this Document

Section 1 - is the introduction to this final report.

Section 2 - provides information on other current public domain research and development in this area.

Section 3 - is a short discussion of the study approach in terms of reductions in systems life cycle cost and testing as a control function.

Section 4 - presents the finding of this research.

Section 5 - is the conclusions

Section 6 - provides the recommendations.

1.4 Testing References

DoD-STD-2167 - Defense System Software Development

DoD-STD-2168 - (Draft) Software Quality Evaluation

TB 18-104 - Technical Bulletin, Army Automation Testing of Computer Software Systems

Mil-STD-490 - Specification Practices

DoD-STD-7935 - Automated Data Systems Documentation Standards

ANSI/MIL-STD-1815A - Ada Reference Manual

AMC-P 70-13 - Army Materiel Command, Software Management Indicators

AMC-P 70-14 - Army Materiel Command, Software Quality Indicators

2.0 OTHER CURRENT PUBLIC DOMAIN RESEARCH AND DEVELOPMENT

This section focuses on representative efforts within the stated study scope of reducing system life cycle costs by using testing as a control feature. This is a succinct, non-exhaustive discussion of current R&D efforts in the software testing metrics, programming environment language domain, and the expert, or knowledge based system domains.

TESTING DOMAIN MATRIX

	Functional Need	A Specification	B5 Specification
Acceptance Testing			
System Testing			
Integration Testing			
Unit Testing			

Figure 2

2.1 Software Testing Metrics

Within of the current state of the art in software testing metrics are two representative metrics; McCabes Cyclomatic Complexity Metric; and Halsteads Information Volume Metric. Software metrics are management tools, scientific/empirically based, that must be unambiguous and objective to be useful. However, metrics are often misunderstood and misapplied.

McCabes Cyclomatic Complexity

- An early attempt to apply the notion of complexity to measure software quality.
- Easy to compute V(G) = Sum (Loops, conditions, cases) + 1.
- Language dependent.
- Very weak when comparing software of same/similar cyclomatic complexities.

Halsteads Information Volume

- Derived from common sense, information theory, and psychology
- Needs automated computation:

  $V = N \log_2 n$, where:

  N = Total Operators and Operands,

  n = Total Unique Operators and Operands.
- Works for any algorithm in any language.
- The major weakness is treating user function calls and system function calls equally.

Other metrics have been successful in specific domains, such as function point analysis. Of course, the old standby metric, is delivered source instructions per person month (lines of code).

2.2 New Environments/Languages

TAME (Tailoring an Ada Measurement Environment) is research ongoing at the University of Maryland. This research aims at the development of a prototype measurement and evaluation environment that supports the measurement and evaluation of the quality, productivity, and product aspects of Ada projects. TAME includes the processes of setting up measurement and evaluation goals and deriving supportive measures. The current prototype does not interface with an (APSE) Ada Programming Support Environment; however, it is designed to be integrated into an APSE in the future. The TAME system provides means for collecting, storing, and validating data, computing measures, and interpreting computed values within the context of particular evaluation goals.

(MOTHRA) Mutation Testing Architecture, is a prototype environment based on the program mutation testing technology at Georgia Tech. The environment is an integrated set of tools and interfaces that support the planning, definition, preparation, execution, analysis and evaluation of tests of software systems. The MOTHRA environment was designed with two primary objectives. The first is that the environment possess high band width user interfaces. The second is that it impose no a priori constraints on the size of software that can be tested in the environment. In addition, it supports the notion of progressive tests, in that it allows the user to carry data from one level of testing to higher levels, with the capability to incorporate that test data into the overall test objectives.

Mutation analysis and thus the MOTHRA environment allows the tester to create test data, evaluate old test data, detect the absence of known errors, and provides error detection. These capabilities are also very applicable to the testing of reusable software. Evaluation of old data and the creation of new test data allows the user of a reusable component to develop test cases that are related to the operational objectives of the system, while eliminating the generation of redundant test data from previous testing.

CSRL (Conceptual Structures Representation Language) has been used by the Battelle Columbus Division for several industrial applications as part of Battelle's contract research business over the last three years.

CSRL, developed by the Laboratory for Artificial Intelligence Research at the Ohio State University, is a language for building classification expert systems. Based on the notion that classification problem-solving can be modeled as a society of specialists, CSRL implements such knowledge-based systems as a classification hierarchy of SPECIALISTs, where individual SPECIALISTS engage in hypothesis refinement, i.e., the task of the problem solver is to find the categories, or hypotheses, within the classification hierarchy which is appropriate to the situation being analyzed. Examples of classification problem-solving include diagnosis, catalog selection, and certain types of planning.

Two major conclusions have resulted from using this language in an industrial setting. First, CSRL is a powerful knowledge engineering tool and it also supports standard software engineering needs for developing computer software. Second, several identified enhancements are necessary to make CSRL a more effective and cost-efficient development tool. With these enhancements, CSRL will satisfy the software engineering goals for the development of expert systems which are testable, reliable, cost-effective, well-documented, understandable, maintainable, and modifiable - software asich meets the user's needs.

2.3 Knowledge Based Systems

SAC (Software Acquisition Consultant) research being performed at the Naval Underwater Systems Center, New London laboratory, CT. The goal of this research is to produce an expert system decision aid for tailoring the requirements of DoD STD 2167, using DoD-HDBK-287, associated Data Item Descriptions (DID), and standards. The current prototype is currently employed in the selection of DIDs to be required for a software development project. The purpose is to provide software acquisition managers, responsible for applying DoD-STD-2167, with software engineering expertise. SAC assists in developing the appropriate level of requirements and documentation tailored for each procurement.

DIOGENES (Expert System for Extraction of Data System Requirements from User Scenarios). This NASA SBIR work derives system requirements from user

scenarios by facilitating and analyzing interactions between software systems engineers and system end users. This prototype system has automated a scenario-driven methodology for deriving top-level specifications and preliminary designs for user data systems.

Expert System for Software Quality Assurance, is a prototype that was created for the US Army Belvoir Research, Development and Engineering Center. This expert system facilitates the process of tailoring statements of work by capturing the knowledge of software QA engineers. The system was executed to alleviate staff turnover/inexperience and to ensure that the consistent standards and requirements of an adequate software QA program are enforced.

ECA (Expert Complexity Analyzer) by Autometrics Inc. is a knowledge based system that provides individual module level analysis, module clean-up suggestions, bug predictions and project scheduling. The system is tailored to the DoD-STD 2167 environment and at the Preliminary Design Stage will provide an initial testing schedule and an effective linear based development schedule without provision for software complexity or defect prediction. At the Preliminary Design Review (PDR) a more refined testing schedule, dynamically based on software complexity, testing personnel allocation, initial defect predictions and resource allocation recommendations is provided. At the critical Design Review (CDR) the system provides schedule and testing personnel allocation reflective of software complexity, full defect prediction for tracking test effectiveness, predicted reliability estimates, test management suggestions, and test management reports. During the Software Testing Phase the system provides analysis of defects found to those predicted, defects remaining estimates, test schedule refinement, test effectiveness reports, and operational availability and system reliability predictions. ECA future capabilities will include; development of a historical project testing database; development of module, unit and build test strategies; development of automated test scenarios; integration of automated test path generation techniques.

ASQ (Automated Assistant for Specification of Software Quality), under development at Dynamics Research Corporation, allows acquisition managers to

cost-effectively specify software quality goals based only on their knowledge of the application and system specifications. The tool provides a mechanism for putting technology to support software quality specification into use in todays's DoD systems. Through automated application of the software quality methodology to DoD systems, the software development community will begin to see the benefits of specifying and measuring quality.

Knowledge-based reasoning allows ASQ to provide the essential software quality guidance to users, and to incorporate prior decisions made by the user for future use. The following are some of the important features of ASQ; Complete, easy-to-use guidance, stepping through each specification procedure, so that the acquisition manager does not need to know details of the software quality specification process; Access to help for users of all experience levels; A flexible menu-driven user-interface; automated procedures wherever possible; Automated extrapolations from available information, whenever a user skips certain details; Incorporation of assumptions and decisions for later review; Incorporation of a database of results from past projects; and specification by example, whenever data from example projects can help specify quality for the current project.

Based upon the above material provided on Testing Metrics, new environments/languages and knowledge based developments, we can see that there is considerable diverse activity in the software testing field. However, we have not found any published reference to a life cycle wide testing architecture to control system development through testing. Our problem statement for this research was; is it feasible today to produce a sound automated software test plan to support DoD system development and test activities? If it is feasible, how would the testing architecture be structured?

3.0 STUDY APPROACH

The approach for this research included five separate tasks: A literature review; Expert knowledge execution; Identification of software testing requirements; definition of a software test case prototype; and determination of feasibility of generating an automated software test plan.

3.1 Literature Review

The first task was to execute a review of the literature and other sources to identify current techniques and tools, including Knowledge Based Systems that were potentially applicable to this study. The second task then focused on reviewing the literature to determine what kind of progress has been achieved in the automatic software test plan generation arena. The focus of the effort was on the identification of any knowledge based systems in existence that are applicable, or that may be modified to be applicable to the problem area.

Whenever applicable systems did exist, they were evaluated in terms of applicability to a particular phase of the process (e.g., most probable error statistics problem) or to the overall Testing Architecture process being researched under this contract.

3.2 Expert Knowledge Extraction

Defining the bounds and the scope of the software test plan generation process was one of the critical tasks of this research. The act of extracting the detailed experience from experts has a magnifying effect upon the typical problem definition. Based upon previous knowledge based system development experience, the problem statement must be very focused because detailed experience brings many "new" factors/sub-problems to bear as a part of the ultimate problem solution set.

Figure 3 depicts the overall "problem space" of the software test plan generation problem at the top of the graphic. The problem space includes the entire undefined testing environment. Although based on Sonex's experience, in testing the current Army Advanced Field Artillery Tactical Data System (AFATDS) development, it was obvious that automatic test data generation was beyond the scope of this research, however, the issue of automated test procedures was less clear. The expert interviews progressed in two stages, first general testing issues, which culminated in the identification of the test plan generation problem as the key. The second stage was the detailed definition of the software test plan generation problem.

PROBLEM DEFINITION PROCESS

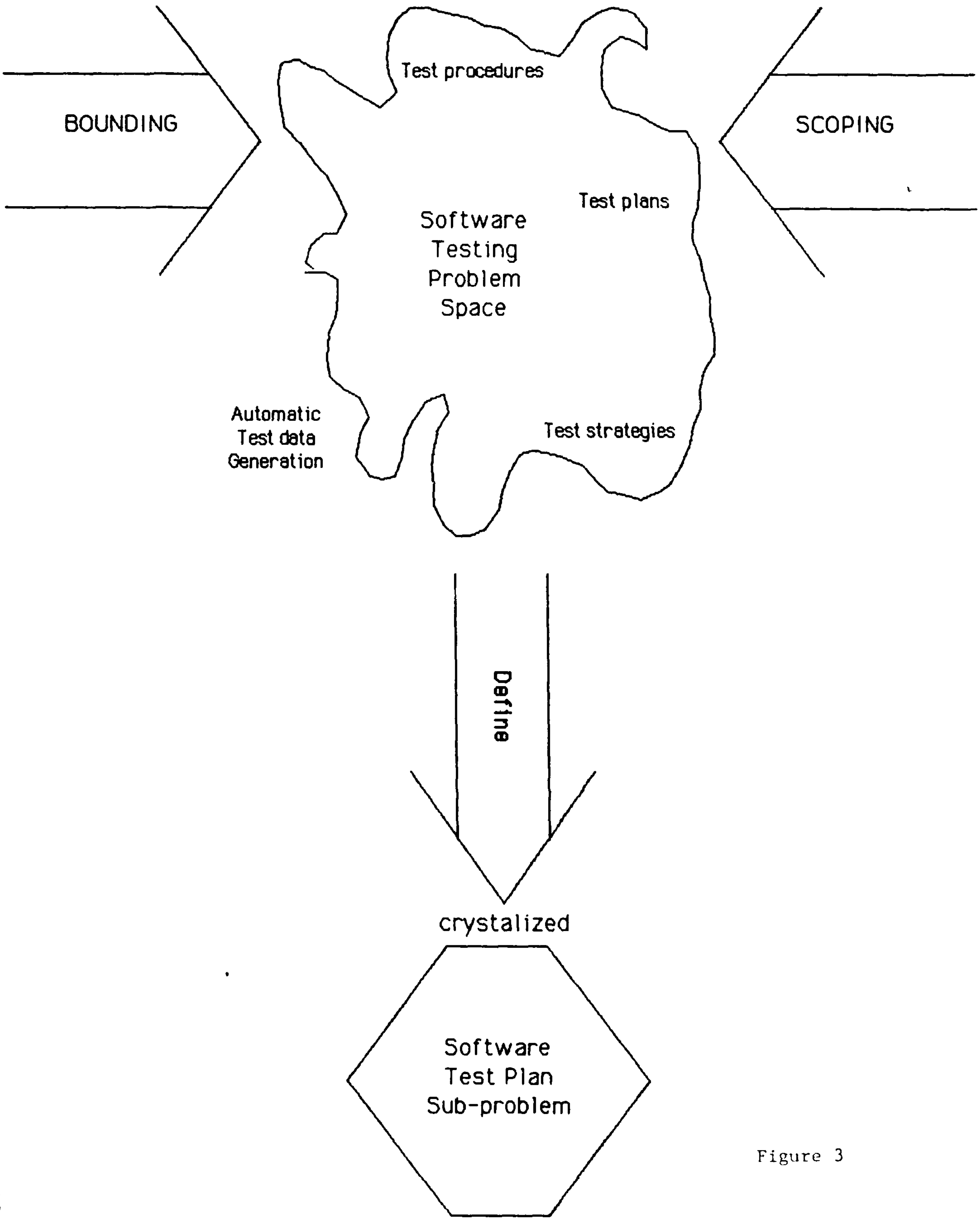


Figure 3

3.3 Software Testing Requirements

After the key problem was identified, defined and approved by the COTR, expert interviews continued with a focus on the Software Test Plan to determine detailed testing requirements. Interview role relationships were defined, the experts were provided information on the research prior to the initial meeting, and the sessions were taped and transcribed. This interview process was complemented with additional research in search of material and methodologies for additional points of view and academic depth.

3.4 Definition of the Test Case Prototype

The initial methodology for the test case prototype was as follows:

- Create a paper system that allows the domain expert to identify and fill the voids.
- Determine the verification criteria for subsequent evaluation.
- Program the system using appropriate Knowledge Based system software.
- Iterate the development with the domain expert to demonstrate various capabilities.
- The prototype shall solve a portion of the vital subset problem suggesting that the approach is viable and further system development is achievable.

However, the magnitude of the task (to define all of software testing) coupled with the requisite thoroughness of the interview process, necessitated an alteration to the research scope. Consequently, the effort was refocused on defining a testing architecture, rather than developing a demonstration prototype knowledge based system.

3.5 Expert System Feasibility

The feasibility of the software testing architecture/Knowledge based system concept was developed and evaluated at the midpoint of the effort.

Specific problems that were expected to be encountered in implementing the system were identified and the methodology used to address these problems was discussed. In addition, the potential for expanding the Software Testing Architecture to include other aspects of Software product assurance was evaluated.

4.0 FINDINGS

4.1 Literature Review and State of the Art Survey

The research consisted of deliberate bibliographic literature searchs by Defense Technical Information Center (DTIC) on AI and Software Testing (at Attachment 1) and the Data Analysis Center for Software (RADC/COED) at Griffiss AFB, NY (at Attachment 2). Interviews with industry and academics in the software development and testing areas were conducted with AT&T Federal Systems, IBM Federal Systems, Boeing Computer Services, IMR Systems Corporation and George Mason University. Sonex has also remained abreast of current technologies through attendance at the Washington DC Chapter of ACM SIGAda meetings, and attendance at representative conferences such as the annual IEEE Expert Systems in Government Symposium, National Conference on Ada Technology and Washington Ada Symposium, and National Institute for Software Quality and Productivity's recent Software Testing and Validation Conference.

From these activities it was concluded that tremendous progress is being made in many diverse fields. However, two separate forces are currently afield: Congress is imposing severe restrictions on defense spending and the spectacular progress in various fields has the potential to create a run-away engine. As Ada, and other new technologies, has been implemented in major software development projects, the impact has been greater than the government or contractors anticipated. Ada has caused both contractor and government personnel to reassess the entire waterfall development methodology. Testing is the only common control that transcends languages, metrics, and methodologies. The needs for test control and discipline have never been greater.

Specific high-level rules of thumb that emerged from this literature review and state of the art survey include:

- Because of sophistication, resource and time constraints, software cannot be tested exhaustively.
- Testing requires mechanization if it is to make a serious impact and control the software development effort.
- The obvious cost benefit of early error detection justifies testing the written requirements for logic errors and determining the testability of the proposed system before any code is written.
- The special problems of real time systems have yet to be solved.
- The advantages of writing the system users manual during the requirements phase, and its use as a testing ground truth.
- Quality cannot be inserted at testing, it must be built into the software product.
- The phenomenon of defect clustering where 80% of the errors are identified in 20% of the code, holds regardless of the PDL.

From the level of actively detected it is evident that software testing is emerging as potentially the most comprehensive control function.

4.2 Experts Interviewed

Mr. Steven J. Callas has over eighteen years industry analysis and management experience including Test Plan development and implementation for hardware and software, customized application development including QA activities, and hands-on experience with IVV and CM.

Increased project productivity by 11% due to project IV&V efforts over two years. Generated test plans for the Army's independent testing of Tactical Analysis software systems and produced a written report describing the results. Evaluated documentation and provided hardware and software configuration management to the Tactical Analysis System. Reported to the Configuration Control Board (CCB) of the project for the preparation and distribution of CCB decisions and supporting data requirements. Created a data base on a VAX 11/780 for maintenance of configuration management

information. Prepared configuration management plans for contract proposals using Department of Defense Standards 7935 and 480; and Military Standards 481, 483, and 490. QA experience using Army TB 18-102 and Navy standards identifying quality factors and how they directly impact the project at hand. Employed QA as a process, not as a checklist. Mr. Callas is currently advising the Magnavox Corporation on testing and configuration management for the AFATDS System development.

Mr. William J. Slobodian has over seven years experience in the software testing, quality assurance and IV&V areas. As Principal Engineer, he has written software test plans and procedures for the US Army Technical Control and Analysis Center (TCAC) and the TOP GALLANT, SIGINT/EW systems of the Joint Tactical Fusion Program.

Duties included the performance analysis of baseline software and firmware. Analyzed detailed software design strategies for integrity of functionality and system architecture. Reviewed all software documentation pertaining to application and vendor supplied software. Performed manpower and cost analysis in relation to software testing and other functions. Served as engineer specialist for C3I Special Projects. Advised the Government Program Manager on software scheduling, hardware acquisitions, and technical aspects of the project.

As Systems/Software Engineer, he had duties including the design, debug and implementation of software for Naval Weapon System WDS MK14. Computer languages used on software project included CMS02M and Ultra-16 for the Sperry Univac AN-UYK20 computer. Work included insuring systems maintenance tests operated according to designated specifications. Served as technical field representative at the Land Based Test Center, Wayland, MA. Specific assignments included system evaluations on launcher and fire-control systems. Prepared monthly reports, trained members of section on operation of OJ-194 and USQ-69 shipboard and AN-UYK 20 operations.

Mr. John S. Williams has a wide and varied background of management of complex projects and supervision of military and technical organizations. He has extensive experience in the definition and development of command and control information systems in a career marked by innovation and achievement of objectives. As project manager for defining the Command and Control Information System (C2IS) for Allied Command Europe, he initiated the use of modern structured methodology which has been adopted as the NATO standard. With direct supervision of a multi-national team of systems analysts and functional experts tasked to define strategic C2IS functional and technical requirements, he presented and justified system requirements before NATO's Technical and Budget committees which resulted in full funding support. erved as Chairman of working groups and committees responsible for the definition of ADP standards, policy and procedures.

As Chief of ADP Quality Assurance, Supreme Headquarters Allied Powers Europe, Mr. Williams conceived, developed and implemented a quality assurance philosophy and procedures which resulted in marked improvement of Command and Control Information Systems in operation and being developed to support SHAPE. Introduced improved Configuration Management procedures and tools.

Mr. Williams currently consults on testing and functional user issues for Magnavox Corporation, the AFATDS system development.

Mr. Michael J. Xenos has 30 years experience in IVV, CM, and QA (including T&E) from which to draw upon. Mr. Xenos conceived and developed the Independent/Integrated Systems Assurance (ISA) concept which integrates QA and CM processes into a structured verification and validation methodology. Pertinent experience includes:

Deputy Program Manager for the development, operations, and maintenance of the US General Accounting Office (GAO) Consolidated Administrative Management Information System (CAMIS), a $23 million nationwide interactive system. He directed the program startup, including securing and orienting the requisite resource, establishing resource accounting policies and procedures, and establishing professional relations with the customer. Negotiated agreements with subcontractors and managed their personnel resources. Managed deliverables to schedule and established the award fee criteria.

Project Manager for development and implementation of major enhancements to the Federal Guaranteed Student Loan (GSL) and the national Direct Student Loans (NDSL) programs ($23 million) under a fixed-priced contract with the Department of Education. He managed both company and subcontractor personnel for development, implementation, and maintenance. Coordinated the telecommunications network and hardware requirements with regional vendors and telephone companies. Represented corporation in weekly Regional Administrators conferences with the Commissioner of Student Financial Assistance. Conceived, orchestrated and supervised the development of the Army's PROBE system that assists DA in the automated development of the annual five-year programs and budgets.

Chief Resource Programs (Manpower, Equipment, Facilities, Dollars) US Army, Europe (1976 - 1977). Mr. Xenos managed a staff of 28 with responsibilities that included: Established the automated functional resources planning structure, established criteria, standards, and procedures for resource planning, initiated development of the European Five-Year Resource Requirements Document, coordinated requirements with NATO headquarters and incorporated US commitments to NATO. Mr. Xenos represented the Command at Headquarters DA resource allocation boards, integrated European resource programs into the Army Five-Year Program and developed requirements for ADP systems to support resource planning.

Adjunct Professor of Management, Computer Sciences and System Analysis, for Boston and American Universities. Developed and taught a graduate level integrated analysis and computer architecture courses for 13 years.

The expertise of these key participants was complemented with short interviews with other testing experts. The result of these interviews and the derived testing architecture are provided in the next section.

4.3 Software Test System Architecture

This section is segmented up into five major sections. These sections discuss the EXTEND (Expert Test Evaluation Node Development) system results of the SBIR Phase I research. The five sections are supported by high level system architecture charts, the EXTEND Execution Flow, Input and Output descriptions grouped by process, a sample session with the prototype expert system shell, and a discussion of feasibility.

4.3.1 EXTEND Overview

The overall system architecture diagram is at Figure 4. The overall EXTEND system architecture chart depicts the main elements of the software testing environment. The Execution flow of the EXTEND system will be discussed in Section 4.3.2. This architecture begins with a subsystem that addresses the potential system target environment and then identifies appropriate Computer Software Configuration Item (CSCI) structures. The proposed CSCI decision tree design impacts significantly on software testability.

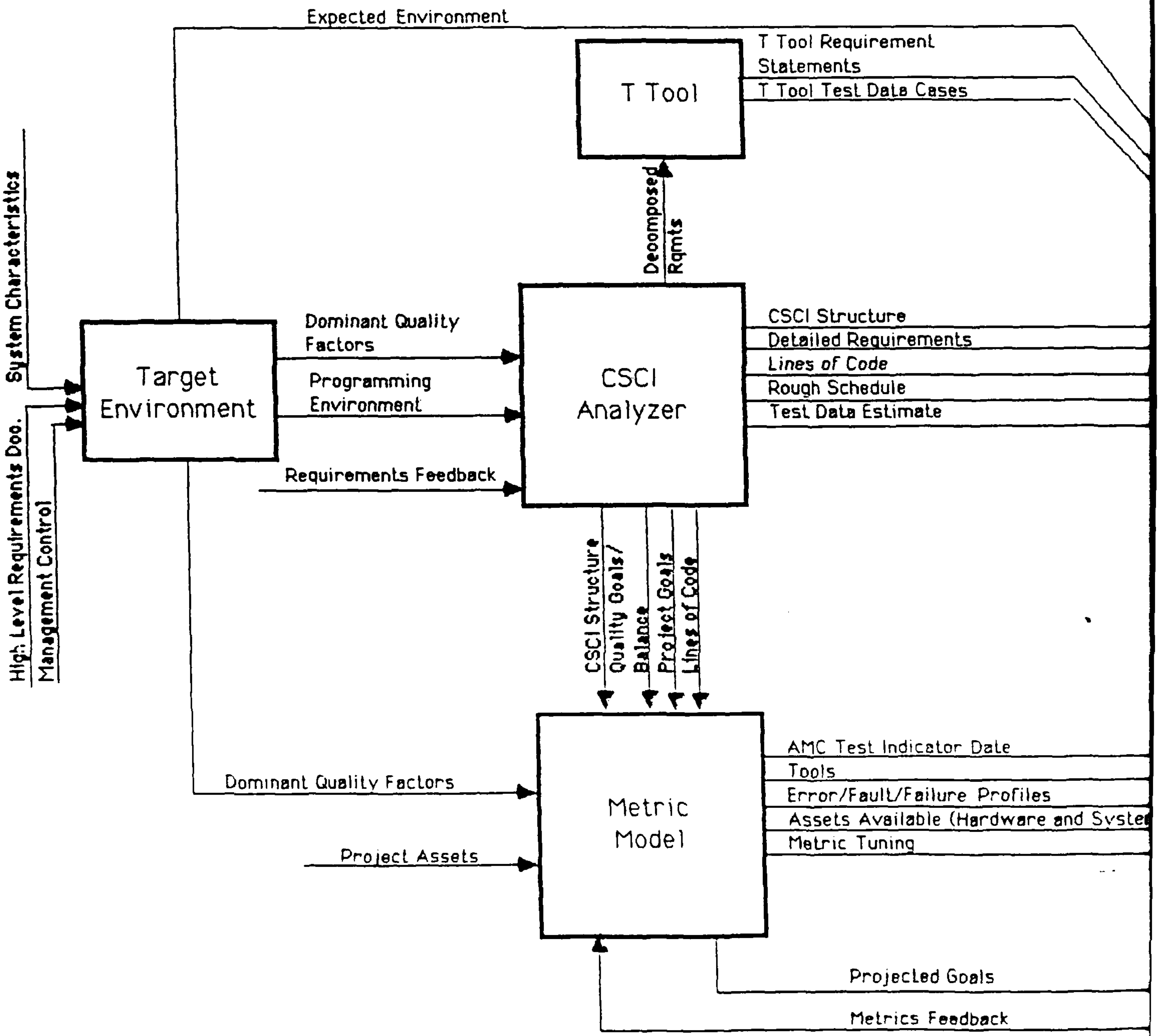
The CSCI tree feeds the Software Test Development subsystem. Other major inputs are Software Metric Model profiles, T Tool (discussed in Section 4.3.3.3) requirements and test data cases. The Test Development Subsystem is the focal item in the EXTEND Architecture. The software test development process is further defined in Figure 5. The software test products of the test development process are provided to a generic testbed or testbed interface and an off-the-shelf tools interface. The final product of the EXTEND architecture is the Test Results Analysis Subsystem that analyzes the tests for acceptance. This sub-system also provides feedback to other components of the EXTEND architecture.

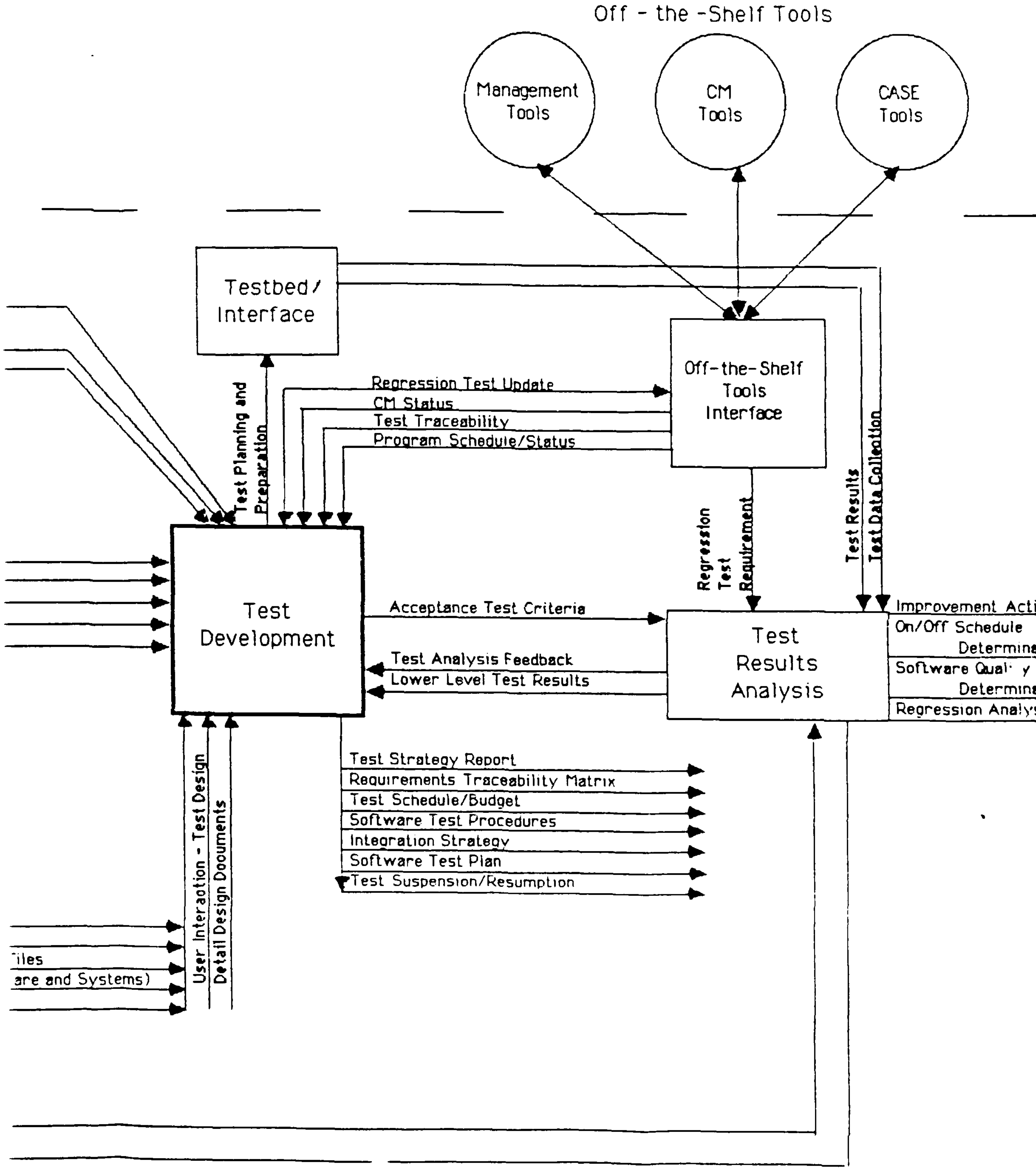
The detailed Test Development Subsystem diagram at Figure 5 is composed of three major functions. The generic flow of control begins with the Test Resources Subsystem and moves to the Test Plans subsystem. Based upon these inputs, the final player is the Test Procedures subsystem.

EXTEND SYSTEM

OUTSIDE SYSTEM

SYSTEM ARCHITECTURE





Off - the -Shelf Tools
Management Tools
CM Tools
CASE Tools
Testbed/ Interface
Off-the-Shelf Tools Interface
Regression Test Update
CM Status
Test Traceability
Program Schedule/Status
Test Planning and Preparation
Regression Test Requirement
Test Results
Test Data Collection
Test Development
Acceptance Test Criteria
Test Analysis Feedback
Lower Level Test Results
Test Results Analysis
Improvement Acti
On/Off Schedule Determina
Software Qual y Determina
Regression Analys
Test Strategy Report
Requirements Traceability Matrix
Test Schedule/Budget
Software Test Procedures
Integration Strategy
Software Test Plan
Test Suspension/Resumption
User Interaction - Test Design
Detail Design Documents
iles
are and Systems)

Figure 4

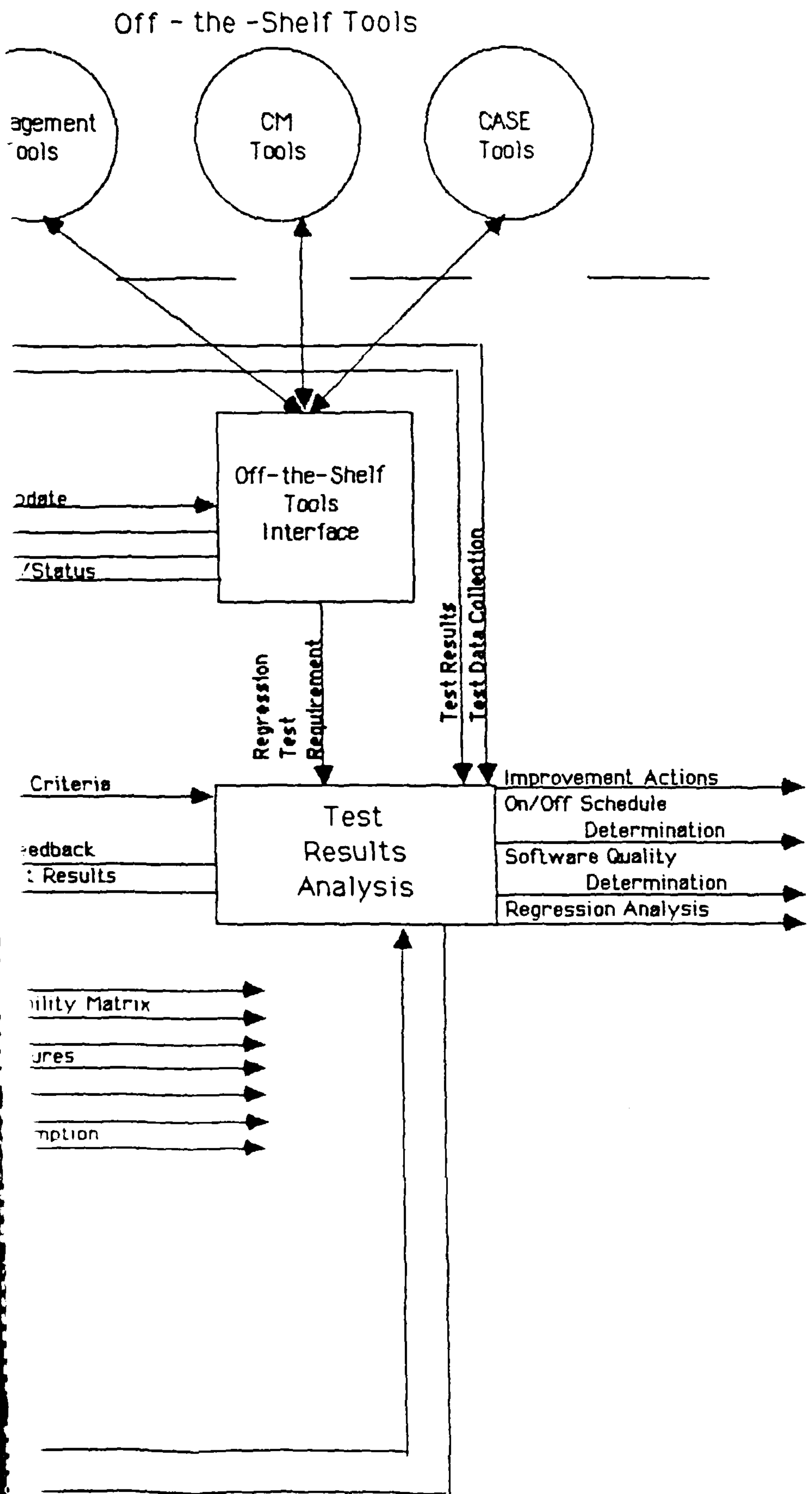
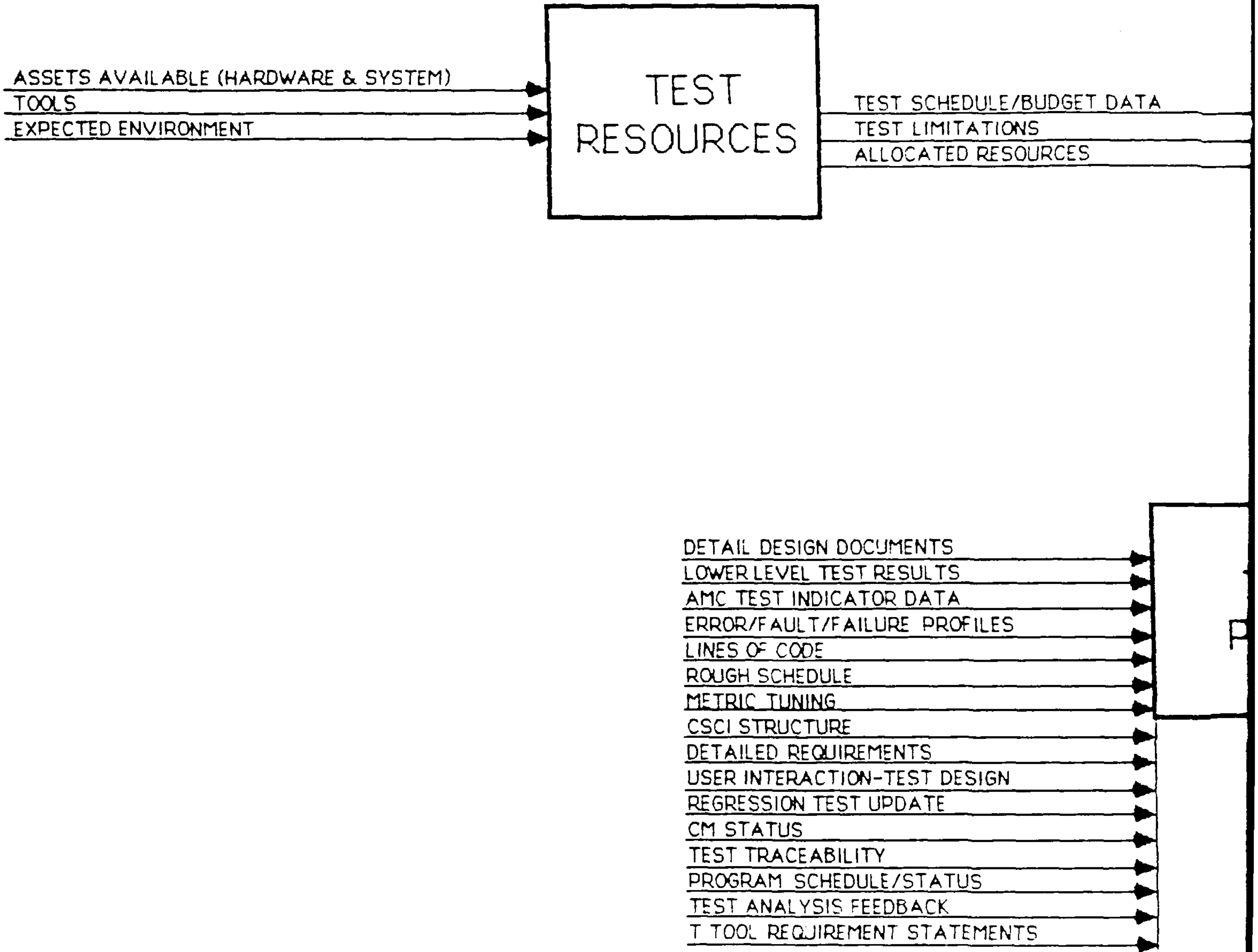

Off - the -Shelf Tools
agement
Tools
CM
Tools
CASE
Tools
Off-the-Shelf
Tools
Interface
odate
/Status
Regression
Test
Requirement
Test Results
Test Data Collection
Criteria
edback
Results
Test
Results
Analysis
Improvement Actions
On/Off Schedule
Determination
Software Quality
Determination
Regression Analysis
ility Matrix
ures
mption

Figure 4

TEST


ASSETS AVAILABLE (HARDWARE & SYSTEM)
TOOLS
EXPECTED ENVIRONMENT
TEST RESOURCES
TEST SCHEDULE/BUDGET DATA
TEST LIMITATIONS
ALLOCATED RESOURCES
DETAIL DESIGN DOCUMENTS
LOWER LEVEL TEST RESULTS
AMC TEST INDICATOR DATA
ERROR/FAULT/FAILURE PROFILES
LINES OF CODE
ROUGH SCHEDULE
METRIC TUNING
CSCI STRUCTURE
DETAILED REQUIREMENTS
USER INTERACTION-TEST DESIGN
REGRESSION TEST UPDATE
CM STATUS
TEST TRACEABILITY
PROGRAM SCHEDULE/STATUS
TEST ANALYSIS FEEDBACK
T TOOL REQUIREMENT STATEMENTS

T DEVELOPMENT

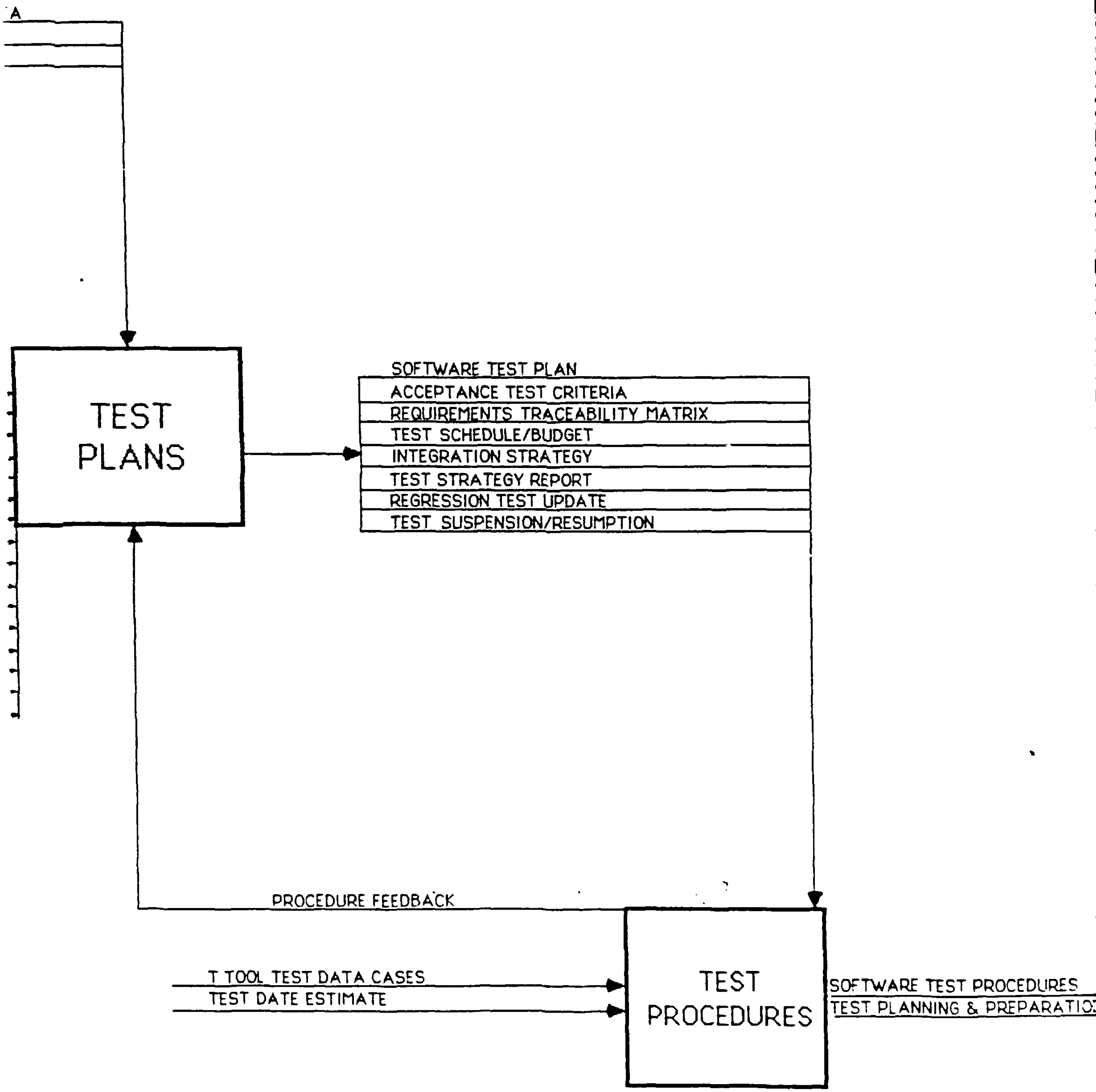


Figure 5

NT

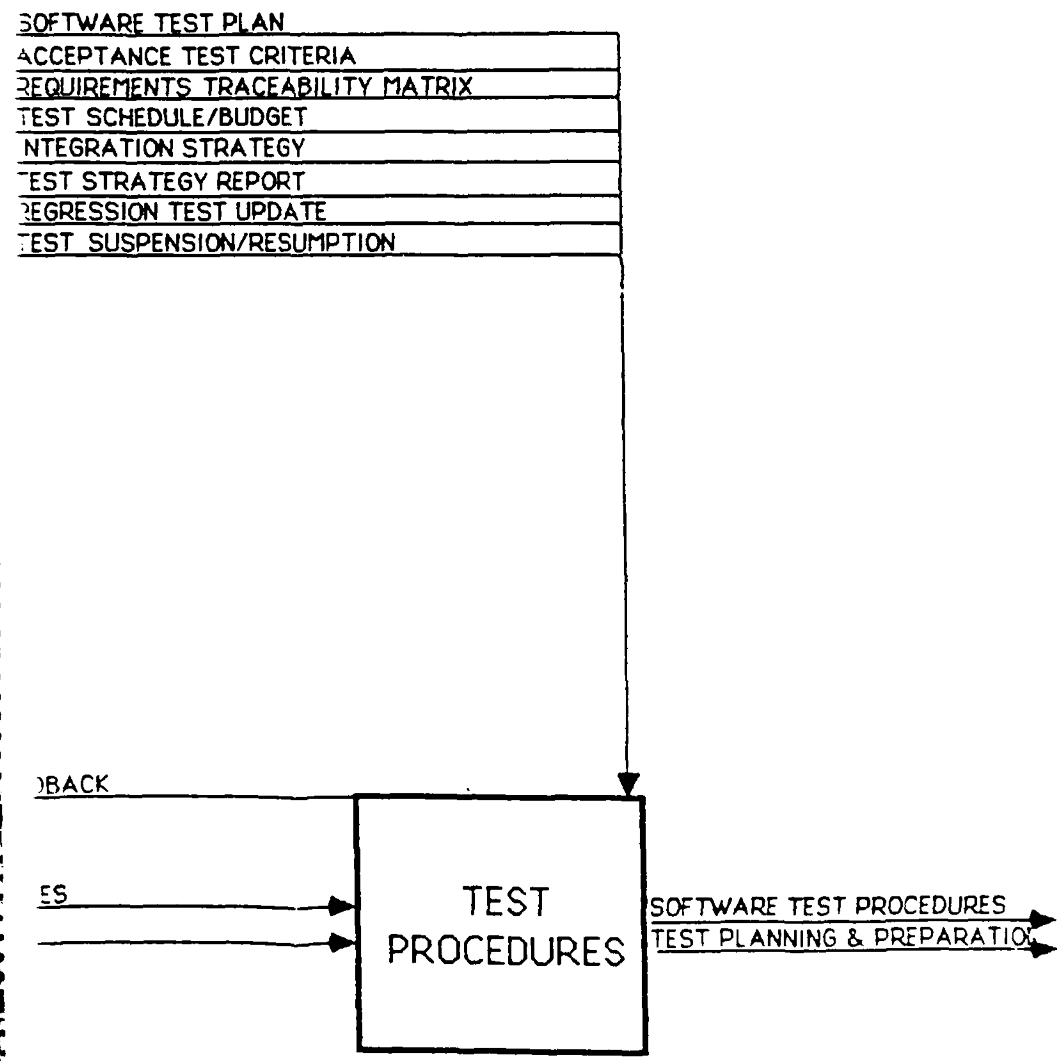

SOFTWARE TEST PLAN
ACCEPTANCE TEST CRITERIA
REQUIREMENTS TRACEABILITY MATRIX
TEST SCHEDULE/BUDGET
NTEGRATION STRATEGY
TEST STRATEGY REPORT
REGRESSION TEST UPDATE
TEST SUSPENSION/RESUMPTION
)BACK
ES
TEST PROCEDURES
SOFTWARE TEST PROCEDURES
TEST PLANNING & PREPARATIO

Figure 5

4.3.2 EXTEND Execution Flow

The EXTEND System architecture has the capability to always produce a software test plan. This software test plan shall be repeatable and one of the software test plan appendices will be the Assumptions and Constraints invoked to create the software test plan. The extent of these appendices is a function of development phase. The Interactive session between the user and the EXTEND system will begin by defining the environment of the target system (see Figure 6). If there is information the EXTEND system requests and the user doesn't know, EXTEND will derive a default value, based on expert experience, rules, and the previous session input. The architecture of the EXTEND system requires a minimum number of data inputs to create the resultant test plan. If the user can not provide the input data, then the interactive EXTEND system module assigns default condition values based upon Expert tester experience assumptions. The basis for these assumptions will come from Sonex research/interviews with testing experts and data bases of historic testing experience. The assumptions for the specific session shall be provided as an appendix of the resultant Software test plan.

Likewise, when the user can answer specific questions about the target environment, then questions about the CSCI structure of the target system will be asked (also depicted in Figure 6). This interaction about the CSCI structure will question the user about the target system requirements and the options for target system design. Based upon the ability of the user to respond, for example, that the detail design data is available, default conditions and assumptions will be made for missing user inputs. These default values and conditions and the resultant assumptions they are based upon, will be provided to the user as part of the software test plan Assumptions and Constraints appendices.

The results from either user input or default values will next activate the Metric Model and T Tool subsystems. The Metric Model subsystem will query the user to extract the minimum data inputs that the Metric Model requires to make a software testing determination. Responses that the user can not accurately detail will have default values associated with them. Again, the default values/conditions will be documented and the supporting assumptions provided as a component of the software test plan Appendices for Assumptions and Constraints.

EXTEND EXECUTION FLOW

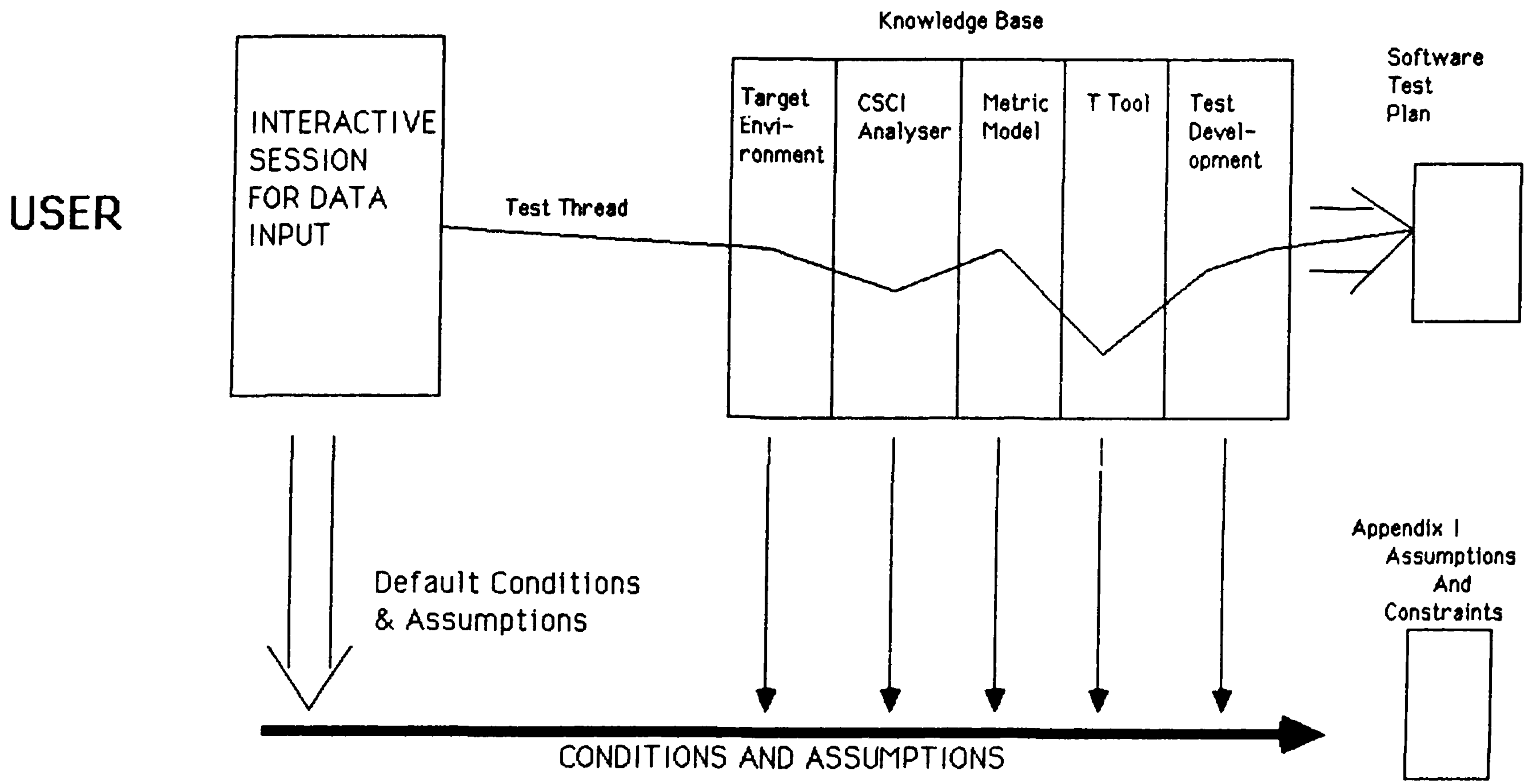


Figure 6

The minimum data inputs, required to create the software test plan came from the user, or through the default condition's mechanism that is generated in the Target Environment, CSCI Analyzer, and Metric Model Subsystems. The appropriate minimum inputs are now provided to the T Tool to generated requirements traceability, test inputs, intermediate test results and expected test results. Additional assumptions and constraints when default values are used will be generated to substantiate the T Tool results.

The result of the T Tool, and CSCI analyzer, and the Metric Model are next provided to the Test Development subsystem to produce the Software Test Plan. The minimum data inputs are provided by the user <u>or</u> the EXTEND system as default values/conditions. These defaults will be provided with backup assumptions that define the default rationale and will be provided as an appendix to the Software Test Plan.

This architecture will produce a Software Test Plan for each formal test. The assumptions and constraints that validate the output software test plan would also be provided as an appendix consistant with DI-MCCR-80014. The EXTEND System architecture will allow the creation of a software test plan, as a living document, during any phase of the target system development. The EXTEND system could be used as the tester progresses through the testing process, allowing the tester to use the EXTEND results as feedback to change his/her original user inputs. The changed/more detailed user input then allows for less EXTEND default values/conditions and provide for a refocused testing strategy. The EXTEND system will always produce the same result with the same inputs. The effects of time, and more available details about the target system, allow for a more detailed software test plan. This more detailed software test plan will generally have less assumptions and constraints attached when the user provides more detail. The EXTEND system could be available to assist in testing decisions, based upon modified user input.

During an update session, the user input about the target system would produce a new software test plan, appendix of assumptions and constraints, and allow a paper (hard copy) comparison of the various sessions. This comparison would also provide the testing knowledge base explanation and rationale for the results.

The EXTEND System's knowledge base would require a very extensive infusion of experience to provide a software test plan. The mapping of the required knowledge and identification of holes in the knowledge will be aided by use of the EXTEND system itself. The subsystem knowledge bases will require updating as holes and voids are identified, and as default conditions and assumptions are critiqued by experts during the EXTEND testing period.

The development of the EXTEND architecture would focus on the establishment of test threads that allow creation of the software test plan. The minimum user inputs or default values to activate the test thread would be identified during the EXTEND analysis. Many possible test threads are depicted in Figures 4 and 5.

4.3.3 Input/Output and Processes

The listings described below provide definition for the terms of processes and data flows (inputs and outputs) used in Figures 4 and 5. The research effort has focused on defining the EXTEND system inputs/outputs as a means of top-down decomposing the overall EXTEND architecture.

4.3.3.1 Target Environment

This process embodies the current target system estimate for the proposed deployment system, including size. The purpose is to identify design and test objectives that affect CSCI Acceptance Testing.

Inputs to the Target Environment are:

1. High Level Requirements Document: For example, the system specification in the Military Standard 490 series documentation. This document must provide system characteristics and identify design constraints <u>if</u> they exist, (i.e., memory size constraints). The requirements are captured in Functional Descriptions, Systems Specifications and "A" Specs.

2. Management Control: The current testing overview with resource, time constraints, and other factors included. Specific areas of interest include data on configuration management, milestones, reporting, budget, logistics, support equipment, facilities, personnel and interoperating systems requirements.

3. System Characteristics: User and designer current estimates of the hardware, software, and operating system for the proposed system. The selection of a DBMS, off-the-shelf requirements, and the development versus the deployment environment issues are also involved. This is working information only. It will change completely as the requirements evolve.

Outputs from the Target Environment are:

1. Programming Environment: The determination of a high order language or assembler, real-time or batch, on-board embedded or special purpose environment. (This input leads to determination of the lines of code). The complexity of the control structure and data is included here. A true lines of code projection is not within the grasp of current technology. However, the intent is to size the software with a measure.

2. Dominant Quality Factors: The identification of key system quality factors. This is developed during a user and system interaction, answering questions on the dominant system characteristics. Examples of system characteristics and related quality factors are: If human lives are involved - dominant quality factors are reliability and correctness; if long system life cycle is critical - testability, maintainability and expandability are the dominant quality factors.

3. Expected Environment: Gross initial estimate of the development and deployment environment (hardware, software, communications, operating systems, etc.) of the target system.

4.3.3.2 CSCI Analyzer

This process executes a Computer Software Configuration Item (CSCI) analysis based upon the requirements. The requirement decomposition is executed through user interaction to a normalized level of requirements. After decomposition multiple recombinations of the sub-requirements are accomplished to check for testability, quality and project goal suitability.

Inputs to the CSCI Analyzer are described as outputs of the Target Environment process with the exception of:

1. Requirements Feedback: Detailed decomposed requirements data as available which is used for system level or test level updates.

Outputs from the CSCI Analyzer are:

1. CSCI Structure: The optimum recomposed structure is identified. The structure will be one of a process (computer process, such as computation, input/output, or table look-up), mission (such as fire control, unit movement, logistics, or communications) or organization (such as Theater, Corps, or Division level) structure tree.

2. Detailed Requirements: The decomposed expanded normalized requirements that form the basis for program specifications, system requirements and specifications are the basis for these detailed requirements.

3. Decomposed Requirements: Low level individual requirements and tables provided in the T-Tool data dictionary format. For example, a generic requirement might be to connect Battlefield node A to B using a required Battlefield conditions table, communications status table and a weather table. This will be a machine readable interface, floppy disk or communications link.

4. Lines of Code: A determination of the "rough" lines of code, either high order language or assembler, without comments compiled. Accuracy will be an order of magnitude rough estimate.

5. Project Goals: Realistic acknowledgment of management goals. Is this a rush, high risk, time sensitive project? Is cost control the key goal? Is schedule compliance the major management goal?

6. Quality Goals/Balance: The quality factors are ordered, expanded, and prioritized. A weighting balance factor is also applied based upon system objectives. If correctness were a key project quality factor -- correctness might for example, be measured through the design parameters, structured methodology, user friendliness, and integration measures. Measurement methodologies and evaluation procedures could then be established for each of these evaluation items. Therefore, one of the design parameters might be capability -- as demonstrated by stress testing.

7. Rough Schedule: A development time table that depicts all the major checkpoints with an identification of the testing constraints/windows to government acceptance testing. The schedule will be un-optimized and not success biased. Plan to estimate the "program design" phase fairly accurately, then move forward in build and install increments.

8. Test Data Estimate: A first estimate of test data requirements based upon the requirements decomposition. This breakdown and identification of detailed requirements (requirements based testing) is the basis for the test data estimate. Specific input data items include input scenarios and operator input messages. The Test Data Collection Requirements captures the results of these inputs.

4.3.3.3 T Tool

T Tool: The T Tool is an off-the-shelf package produced by Programming Environments, Inc. (PEI) and will be integrated into this architecture. The T Tool is a PC based software tool that automatically designs, generates, traces, and documents software test cases from a system requirement. It has proven effective in cutting time and costs from development and maintenance schedules while improving quality. All functions are covered, most probable error coverage includes samples from all coverage areas, and very high structure coverage is provided with the T Tool. Users direct the T Tool through an adaptive interface that varies from menu-based to command-line according to its usage. Prompts, memory aids, and help messages guide the user at all three easy steps. The user enters software descriptions into several different fill-in-the-blank screens: a requirement statement screen and screens for data, condition, event, and state definitions. A restricted English sentence structure is provided on all screens. The user never has to worry about sentence, paragraph, or document structure or how to position the cursor. The T Tool automatically checks all dictionaries and requirements for completeness and consistency. The T Tool automatically produces a requirements specification that can be printed or viewed on-line and included in other documents.

Inputs to the T Tool are described as an output of the CSCI Analyzer process.

Outputs from the T Tool are:

1. T Tool Requirement Statements: This data flow includes the system requirements encoded in the T Tool data dictionary language. These requirements and test cases are blind to regression and stress testing, and do not have horizontal or hierarchical links. These requirement links must be established by other inputs.

2. T Tool Test Data Cases: The results of the T Tool analysis are based on test case requirements statements. Emphasis is on test coverage and

test case productivity. The T Tool generated test cases will exercise at least once, every requirement for 100% function coverage and every most probable error coverage for 100% coverage. Every requirement and every most probable error category will be addressed at least once.

4.3.3.4 Metric Model

Metric Model: This process uses historic defect profiles to characterize the development environment. The purpose is to evaluate project goals, address effectiveness of testing methods, and evaluate testing tools in a quantitative measure. This system is executed through choosing QA and testing methods and tools that fit the input characteristics, interactively evaluate the system behavior and refine goals based upon evaluation results. Testing metrics included are Cyclomatic Complexity, Information Volume, Function Points and others.

Inputs to the Metric Model are described as outputs of Target Environment and CSCI Analyzer. Two additional inputs are:

1. Project Assets: An identification of the software testing assets available to testing. This includes tools, personnel skills, resources, time, hardware and system assets.

2. Metrics Feedback: A result from the Test Results Analysis subsystem. This is an interpretation of the testing results that is used to tune the software metric model criteria.

Outputs from the Metric Model are:

1. AMC Test Indicator Data: Army Materiel Command (AMC) input for the software progress-development and test management indicators of AMC Pamphlets 70-13 and 70-14 will be provided. The quality indicators and time/schedule metrics and their effects on testing will be the key areas of interest. These indicators will be tailored for the software test description.

2. Assets Available (Hardware and Systems): A delineation of projected computer assets that are to be used in the test strategy generation, including testing mechanization assets.

3. Error/Fault/Failure Profiles: Quantification of expected fault detection, error prevention, error profile, and failure data. Methods include functional testing, structural testing, and code reading. Tools include chief programmer team, document library, code reuse, and program design language. Fault type examples include control, data and interface. Error classes include application, environment or clerical type examples.

4. Metric Tuning: This is feedback on the use of current software testing metrics results. This input addresses specific recommendations as a minimum for integration, acceptance, and test procedure modification. Metric tuning also includes test indicator data to indicate quality factors for data base development, scheduling metrics and efforts on raw testing. These indicators will be tailored to the software test plan. Most probable error statistics data is also generated by the Metric Model and tailored to the software design/development methodology (i.e., OOD, Top-Down) and Language(s) of implementation (i.e., Ada, Fortran, Assembler).

5. Projected Goals: Based upon the project environment and monitoring the methods and tools for testing, projected goals that affect the project outcome are recommended. A question, metric, and goal paradigm is used to interpret results and provide a framework for iterative refinement through feedback.

6. Tools: Specific software code or programs to be used as recommended by test metrics, resources, time and other modeled factors. The most important component is functional expertise. Quality Assurance (QA) must have knowledgeable "user" personnel to ensure the "right" system is built. Identification of the best fit Quality Assurance tools is critical to monitor system development. Ensuring the relevance and

balance of the QA reporting and compliance mechanism is a tool responsibility.

4.3.3.5 Test Development

Test Development: This subsystem (process) is composed of three major components: test resources; test plans; and test procedures. These components have been the emphasis of this research. See an expansion of this subsystem in the Test Development Subsystem detail chart attached at Figure 5.

Inputs to Test Development are the outputs from the T Tool; CSCI Analyzer; and the Metric Model previously identified. Additional inputs include:

1. User Interaction - Test Design: This dataflow is a dialog between the user and the EXTEND system. The system architecture requires a minimum number of data inputs to create the resultant test plan.

2. Detail Design Documents: These document will include as a minimum the following; Target System Operators/User manual, Target System size and complexity data, system and program design details, other test assets available, integration strategy baseline, and software builds documents.

3. Lower Level Test Results: This is the result of previous level testing (system, acceptance, or user). This is used within the Test Design Subsystem to match with regression tests and regression test data to test the integrity of lower level test results.

4. Test Analysis Feedback: The feedback is a critical analysis of the test results and test data collected from the previous series of tests. The dataflow response in a timely way allows for tailoring of the test design to analyzed problem areas.

5. CM Status: From a Configuration Management (CM) interface subsystem that links with an off-the-shelf CM tool, like Expertware's CM Toolkit. CM

status provides the test bed status, results, and system control informtion. Also provided is problem reporting, cross reference, version description, and build specification tool status. Configuration Identification and Control is maintained for software, hardware, interfaces, support equipment, instrumentation, data bases, and all diagrams.

6. Program Schedule/Status: A user update from an established software development model like the Putnam Cocomo model, the design element, or others. A schedule for the subject development effort is provided. This input may come from a tool which may be an off-the-shelf component interfaced with the system. A management input, separate from a model, with real subject project data will also be a component of this input. The program schedule/status also includes a design update interactive dialog that provides for the off-the-shelf interface to system update design decisions and constraints.

7. Regression Test Update: The CM input for regression testing including, as a minimum, the test version and test build, test data, and thread control. It includes previous test input, output, and automatic verification, and may include a key stroke saver function.

8. Test Traceability: From the off-the-shelf Tools Interface subsystem to the other Computer Aided Software Engineering (CASE) tools used in the development. This may range from a spreadsheet to a data base to a project management or CM tool to control the testing traceability and control.

Outputs from Test Development process are discussed relative to the subordinate Test Devlopment Subprocesses; Test Resources, Test Plans and Test Procedures.

4.3.3.5.1 Test Resources

This subprocess accomplishes the identification and utilization of all available testing assets. Time and schedule issues must be analyzed with

knowledge of test design and system requirements.

Inputs were discussed as inputs to the Test Development Process.

Outputs from Test Resources are:

1. Allocated Resources: The results of a feedback function between test resources and test plans and procedures subsystems that perform a what-if analysis of the existing resources. Specific input items include schedule, personnel, software, instrumentation, hardware, test drivers, facilities, and interfacing systems data.

2. Test Limitations: Prioritization of all testing features. Features or significant combinations of features that cannot be tested will be identified. A reason will be provided with a probable risk assessment.

3. Test Schedule/Budget Data: A critital path type testing schedule that addresses time, resources and priorities.

4.3.3.5.2 Test Plans

This subprocess performs the quantification and mapping of all test issues and criteria, test cases, test strategy and the kind of test to the system requirements. The system requirements include both operational and functional specifics such as performance, response, and capacity. Based upon user interaction, software test descriptions and software test procedure documents will be outputs. The Test Plans process is broken into the following sub-processes: input correlation, requirements analysis, test planning, knowledge based processing, user interaction, and test report generation. Components include Test Strategy and Test Issues and Criteria sub-subprocesses. Test Issues and Criteria: This process identifies the specific test issues, based upon the CSCI structure, system requirement and system design whose impact should be addressed by the test strategy. The test criteria determination is provided as test case input to test strategy and integration planning.

Inputs to Test Plans were described as inputs to the Test Development Process except:

1. Procedure Feedback: This is an internal evaluation loop within the Test Development subsystem.

Outputs from Test Plans are:

1. Acceptance Test Criteria: The specific user criteria to allow a determination of the system capability to support the functional requirements. The form and measurement of the criteria will be defined and quantified. Risks will also be defined and quantified. Based upon the stated acceptance test criteria.

2. Integration Strategy: This will be a component of the software test plan showing values covering, for example: integration testing, time, and requirement functionality testability. A project unique integration strategy balance of testing must be identified. Integration Strategy Baseline is based on system criticality, software design and testing approach. Critical components and interfaces must be formally tested. The high or low impact of the software failing must be acknowledged. The update is appropriate to development and qualification testing. The result is only a recommendation, which may require a contract modification to implement on the subject system under development. This dataflow defines the orchestration of integration, system, and acceptance testing. Also identified is who performs what functions, such as approving the results, of each test.

3. Requirements Traceability Matrix: A matrix of the detailed requirements to the test strategies. The sequence, level of detail, complexity, limitations, and other parameters will be addressed for each detailed requirement.

4. Regression Test Update: Interdependencies are identified of the tests designed, analyzed and reported. Test cases from all prior tests may be

repeated, and new ones identified. Regression testing validates new or modified requirements that necessitated change, while ensuring existing requirements have not been invalidated. Regression Testing is an iterative process that ensures the testing baseline is not corrupted.

5. Software Test Plan: This data flow is the result of the Test Plan and Procedures Subsystem. The Test Plan follows the formt of Data Item Description DI-MCCR-80014. However, the format shall be flexible enough to readjust as format changes are determined by the user.

6. Test Schedule/Budget: A recommendation from the test strategy viewpoint of where testing emphasis should be placed. An acknowledgment of the impact on testing of schedule, resource, and technical risks attendant to the specific ongoing development. The emphasis is on testing time and the risk effect on testing coverage and the telescoping of any test slippages. The test schedule provides tradeoffs of increased risks (short cuts) to the requirements and quality factors. As Fred Brooks has stated, "More software projects have gone awry for lack of calendar time than for all other causes combined."

7. Test Strategy Report: Recommendation of the testing mix, with percentages for the current testing. This includes code walkthroughs, code reading, document reviews, black box and glass/white box testing, performance testing, etc. The test approach will be delineated for unit, system, integration and acceptance testing. Software build testing status and relationship with all key development milestones will be addressed.

8. Test Planning and Preparation: All the information required, in a "generic" mode, for test running on the subject system testbed. This includes the specific test parameters, environment, procedures and data for a specific test.

9. Test Suspension/Resumption: This dataflow identifies the criteria used to suspend all or a portion of the testing activity. The resumption element specifies the testing activities that must be repeated when testing is resumed -- and any preconditions for resumption.

4.3.3.5.3 Test Procedures

The test procedure development that describes all inputs, outputs for the System testing and produces the Software Test Procedures. Inputs were described as inputs to the Test Development Process. The Output from Test Procedures is:

1. Software Test Procedures: The test procedures that are executed along with the test data, test time, and other input to be run on the target system or testbed system.

4.3.3.6 Testbed/Interface

Testbed/Interface: This process executes the preparation and formatting of all the data required to allow the physical running of tests on the subject development system testbed or the target system if available. This process result may be provided in a machine readable format or through a telecommunication/network type link.

The input to the Testbed/Interface Process was described as an output of the Test Development process the Test Planning and Preparation data flow.

Outputs from Testbed Interface are:

1. Test Data Collection: The total data collection from the subject test. This includes the original test planning and preparation input from the test design subsystem.

2. Test Results: The total results of the subject test, in a machine readable format.

4.3.3.7 Off-The-Shelf Tools Interface

Off-The-Shelf Tools Interface: This interface process allows for the interaction of the EXTEND Testing system and currently developed standard product offerings of Configuration Management, Computer Aided Software Engineering, and schedule/resource management tools.

The input to Off-The-Shelf Tools Interface, the regression test update dataflow, was previously described as an input to the Test Development process.

Outputs from Off-The-Shelf Tools Interface have been previously identified as inputs to the Test Development process and also include:

1. Regression Test Requirement: The CM input including the test version and test build, test data and thread control information.

4.3.3.8 Test Results Analysis

Test Results Analysis: This subsystem (process) compares the test procedure expected test result with the actual test result. It verifies that regression test requirements were included. The results of this analysis provide the feedback to the other subsystem to tune the testing effort.

Inputs to the Test Results Analysis Process are described as outputs of the Off-The-Shelf Tools Interface, Testbed/Interface, Metric Model, and Test Development Processes.

Outputs from Test Results Analysis are:

1. On/Off Schedule Determination: A simple determination of testing proceeding on or off the testing schedule. The determination will be explained with specific problem areas.

2. Software Quality Determination: An analysis of the current development ability to test and verify the dominant quality factors.

3. Improvement Actions: The actions available, with resource and time components, to improve the testing effort.

4. Regression Analysis: A determination that the subject system development integrity has been validated and that the development process has not invalidated the baselined system components.

4.4 Prototype Software Testing Expert System

The following section is a sample transcript from the prototype expert system shell. The purpose is to show the highly user interactive nature available and the level of user assistance resident in the shell. A rephrased question, summary of the current consultation (dialog), partial conclusion's list from the expert system, and the ability to back-up to the previous question are all options available to the user.

An expert system shell was used to assist in prototyping the interactive session of the EXTEND system for user testing data input. The shell is Sonex software written in the MULISP (Soft Warehouse, Inc.) implementation of LISP. The shell models a decision tree structure. The shell uses a modified "If A Then B Else C" rule form.

Results. This initial prototype software testing expert system effort had no architecture and no direction. The prototype development had nothing substantial to relate itself with until a testing architecture had been defined. Therefore, the decision was made to halt this research thrust and continue with expanding the overall architecture displayed in Section 4.3 of this report.

* * * SOFTWARE TESTING ASSISTANT SYSTEM * * *

To evaluate the software project characteristics, design methodology and software reuse factors to access most probable error statistics and define acceptance test criteria.
In using the program you will be asked to answer questions about the existence of various kinds of evidence. Answers to questions can be "Y", "N", or if you dont know, "D". To quit, enter "Q". In addition to supplying answers, you can request information at any time with the following commands:

```
? -- Print a rephrased version of the question
A -- Access database to assist in answering the question
S -- Print a summary at this point in the consultation
T -- Let's you see EXTEND's partial conclusions
U -- Stops the trace
W -- Show answer to a previous question
B -- Break the consultation and put you in MuLISP
```

Press RETURN to continue ...

EXTEND SOFTWARE TEST PLAN GENERATOR

project data options

Number	inputs
1	PROJECT-CHARACTERISTICS
2	TEST-PARAMETER-DESIGN
3	SOFTWARE-REUSE-FACTORS

```
Enter number or Q to quit:
2
 The following is intended to recommend test parameter design criteria tailored
 on your situation

 1 -- For which of the following do you have any information:

 1) Size of the development
 2) Design Methodology
 3) Other Factors

 Please enter one or more of the preceding numbers

      (separated by blanks and terminating with a <CR>  1
2
3

 Please think in terms of defining development size as large, medium or small.

 2 -- Is this project a small development? (Y/N/D/A/S/T/U/W/Q/B/?) :
N
 3 -- Is this a medium sized development? (Y/N/D/A/S/T/U/W/Q/B/?) :
Y
 4 -- Software development type is a Command-
Control-Communications-Intelligence C3I system? (Y/N/D/A/S/T/U/W/Q/B/?) :
N
 5 -- Software development type is a Communications type system?
 (Y/N/D/A/S/T/U/W/Q/B/?) :
N
 6 -- Software development type is a Field Artillery type system?
 (Y/N/D/A/S/T/U/W/Q/B/?) :
N
 7 -- Software development is an Intelligence/ Signal type system?
 (Y/N/D/A/S/T/U/W/Q/B/?) :
Y
 8 -- For which of the following do you have any information:

 1) Object Oriented Design
 2) Yourdon-DeMarco Design
 3) Top Down Design
 4) Middle Out, Iterative
 5) Combination Design

 Please enter one or more of the preceding numbers
      (separated by blanks and terminating with a <CR>  3


 9 -- Are there other factors, besides size & design Methodology, that effect
this development? (Y/N/D/A/S/T/U/W/Q/B/?) :
Y

 ------------------
```

Based on your responses, my evaluation is as follows:

1) Very exacting controls required on SIGINT development

2) Standard design, testing better not be top-down. If testing is also top-down then give special IVV and Govt. personnel attention.

3) Evidence of project to include the requirements of AMC-P 70 -13 and 70 -14

You have established:

4) Size of the development
5) No small development size
6) medium system development
7) No c3I development
8) No comm system development
9) No field Artillery system development
10) SIGINT system development
11) Design Methodology
12) Top Down Design
13) Other Factors
14) Identify other factors

Do you wish to consider another inputs or another topic (Y/N/Q/B/?):
N

4.5 Estimate of Architecture Feasibility

The subject of software testing has not been an area of major architecture research of expert systems application. There are multiple probable reasons why this is true. The principle reason is that software testing is not generally considered an end upon itself. The typical software developer looks at software testing as a necessary evil, not an integral part of the system development process. We believe this mentality that testing is only a "drain on development resources" is very pervasive. This negative climate has created the vacuum of significant architecture expert systems work in the software testing domain.

We believe the EXTEND System is feasible, because the system components are feasible. Refering back to Figure 6, the EXTEND Execution Flow, the key components of the system are the separate subsystems, the user interface, and the definition of the testing threads. The EXTEND system consists of the Target Environment, CSCI Analyzer, Metric Model, T Tool, the Test Development Testbed/Interface, Off-the-shelf Tools interface, and Test Results Analysis Subsystems.

The Target Environment is a modest effort that begins the target system definition, and provides default values/conditions and assumptions when the user can't provide specific data. The CSCI analyzer defines the Target System software development structure options and has commonality with the DIOGENES system discussed in the Section 2.0 Other Current Research/Developments. Our implementation will build from DIOGENES lessons learned and may integrate a portion of the DIOGENES system knowledge base/logic structure. The Metric Model subsystem is based upon current work ongoing at the University of Maryland, the TAME (Tailoring and Ada Measurement Environment) system. The EXTEND system development will use germaine knowledge base logic and structure from the TAME system.

The T Tool is produced by PEI. The EXTEND development will directly interface and include, the T Tool.

The Test Development Subsystem would provide the EXTEND System final output of the software test description. This component has no direct precident that we are currently aware of. The Sonex approach was to focus

on the Test Development subsystem for the Phase I SBIR research. This continued decomposition of the test development subsystem has produced a better understanding of the total EXTEND knowledge requirements.

The generic feasiblity input-process-output diagram is depicted in Figure 7. The outputs identified in Figure 7 are now discussed in terms of the EXTEND system architecture.

Identification of most probable error statistics: This will be accomplished within the Metric Models subsystem of the EXTEND architecture. This data will be based upon the research provided in the University of Maryland TAME project. This measurement and evaluation environment will provide probable error statistics. Based upon user input, target environment, and CSCI analysis user inputs or default values, the conditions will be generated. These values will be passed to the T Tool and Test Development Subsystem, and the default conditions and assumptions provided in the software test description appendix of assumptions and constraints. This feature will be implemented in the Phase II SBIR development.

Definition of Acceptance Test Criteria: The design of the acceptance testing at the CSCI level shall be the major focus of the EXTEND System Phase II development. The software test description that is generated, the experts contacted and the knowledge extracted will focus on the CSCI Acceptance level testing.

Integration Plan: This will be partially implemented through the resultant software test description and constraints. The effect of different user inputs and the resultant generation of default values will provide an ability to compare integration plan strategy results as depicted within the software test description, and the captured experience of experts.

Identification of multiple Test Strategies: This will be partially implemented in the Phase II SBIR EXTEND development through the software test description product. The different user inputs, and the built-in default conditions and strategies will be executed. The criteria and conditions that identified a specific strategy will be captured in the software test description appendix of assumptions and constraints.

FEASIBILITY PROCESS DIAGRAM

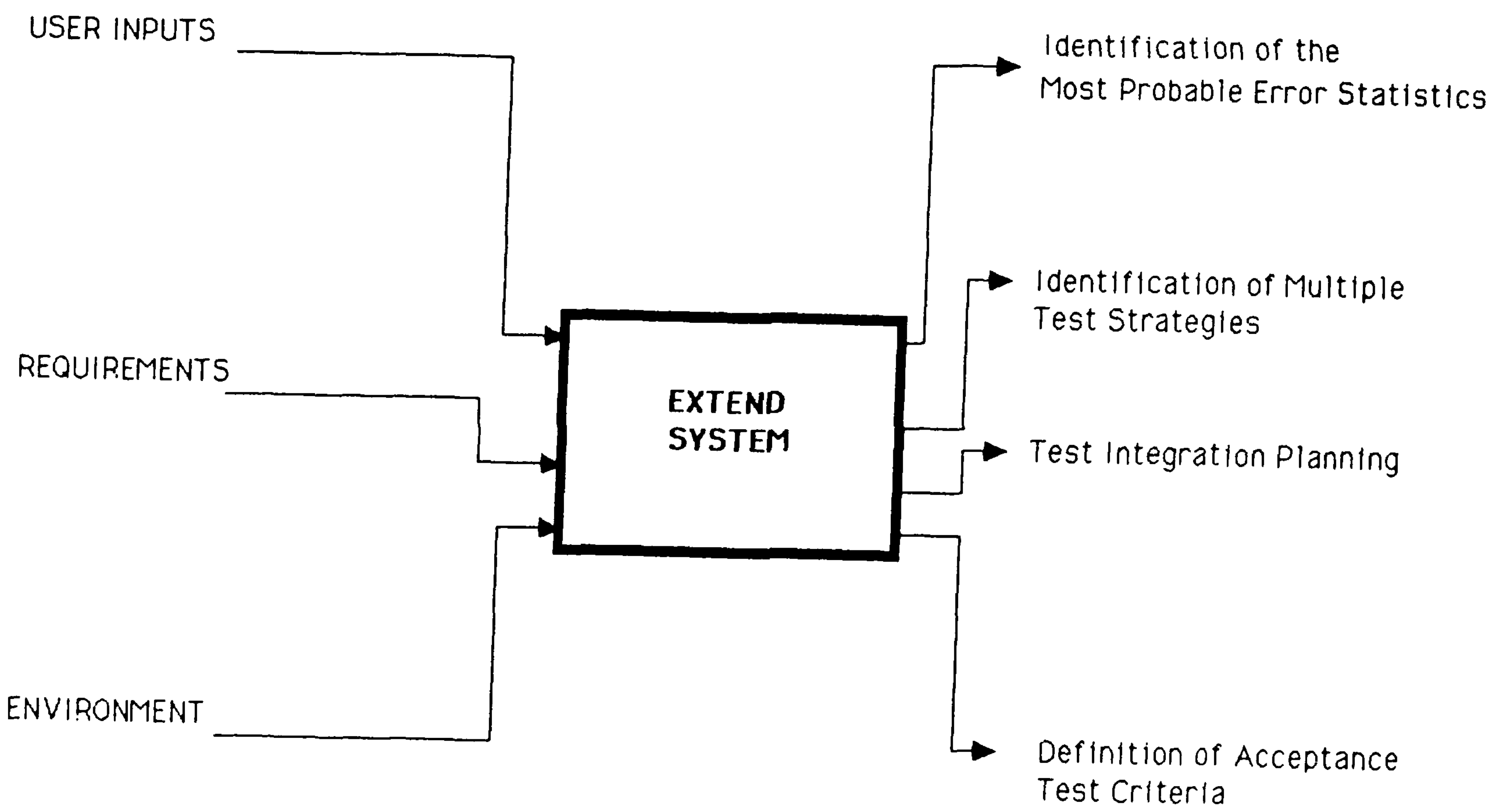


Figure 7

5.0 CONCLUSIONS

The conclusions fall into two categories, conclusions on testing and conclusions on knowledge based system development.

Testing. The problem definition stage of this research was the hardest task. We believe the testing field is very splintered, but generally accepted engineering principles do exist. The practitioners in the testing domain are artisans who do what they do because it has worked before. These testers find it hard to explain why. The testing domain uses reasoning by analogy to a great extent. There seems to be global commonality of overall testing approaches. Certain truths are acknowledged as discussed in Section 4.1.6 Results of literature reviews and state of the art survey. However, an architecture to tie the testing and control functions together in the software development process, is missing.

Knowledge Based Systems. Artificial Intelligence and Expert Systems have been the focus of much recent activity. There is value in the knowledge based system development -- but it is a difficult process to implement. There are two major hurdles in knowledge based system development: The scope definition and the expert knowledge extraction problems. We have addressed the execution of testing from a global point-of-view. Our approach for addressing the knowledge based development is as follows:

1. We have identified the key component of testing, and labeled it the Test Development process in the EXTEND architecture.

2. Address first and further define a well understood smaller segment of the Test Development process, the Test Plan subprocess.

3. Define the environment for the Test Plan subprocess.

4. Structure the environment through the use of logic trees, Data Flow Diagrams, and other techniques.

5. Identify one very senior testing expert to critique the representation of the Test Plan subprocess.

6. Iterate the above approach until all the subprocess facets have been addressed.

7. Continue to refine the detail in the logic trees and other representations with detailed domain facts from testing experts.

8. Identify the voids and inconsistencies in the compiled knowledge.

9. Identify the "best" or most knowledgeable expert to fill the identified voids in the testing knowledge.

10. Execute all the above activities in parallel with user interface prototyping.

6.0 RECOMMENDATIONS:

The findings of the Phase I effort justify the recommendation that the EXTEND system be implemented as a Phase II of the current SBIR program. The recommended approach includes:

- Implementation of a full scale knowledge engineering effort to develop a prototype based on the Test Plan subsystem.
- Focus of the implementation on the key testing issues, for example, metrics, test strategies, and automation assistance.
- Interface with the T Tool and other CASE tools to capitalize on off-the-shelf and ongoing development efforts.

DEFENSE TECHNICAL INFORMATION CENTER

DEFENSE LOGISTICS AGENCY

TECHNICAL REPORT SUMMARIES

ATTACHMENT 1

SEARCH CONTROL NO. (058100 87/03/20) -- BIB

AI AND SOFTWARE TESTING (U)

27847
TO: SONEX ENTERPRISES INC
PRESIDENT'S OFC
VIENNA, VA 22180-

REQUESTED BY: D MATCHINSKI/893-5988 3/20/87

THIS REPORT (COMPILATION) IS MARKED TO REFLECT THE HIGHEST CLASSIFICATION OF ANY COMPONENT PART THEREOF. IT SHOULD BE DESTROYED WITHIN 12 MONTHS AS THESE RECORDS ARE UPDATED AT LEAST ANNUALLY.

UNCLASSIFIED

(THIS PAGE IS UNCLASSIFIED)

WARNING -- This Report May Be Subject To Export Control (See Reverse Side)

NOTICES

****

THE FOLLOWING APPLIES IF THIS BIBLIOGRAPHY INCLUDES ABSTRACTS WITH REFERENCES MARKED "EXPORT CONTROL"

WARNING

THIS DOCUMENT CONTAINS TECHNICAL DATA WHOSE EXPORT IS RESTRICTED BY THE ARMS EXPORT CONTROL ACT (TITLE 22, U.S.C., SEC. 2751 ET SEQ.) OR EXECUTIVE ORDER 12470. VIOLATIONS OF THESE EXPORT LAWS ARE SUBJECT TO SEVERE CRIMINAL PENALTIES. DISTRIBUTION OF THIS DOCUMENT IS SUBJECT TO DODD 5230.25 PROCEDURES.

****

DESTRUCTION NOTICE

FOR CLASSIFIED DOCUMENTS, FOLLOW THE PROCEDURES IN DOD 5200.22-M, INDUSTRIAL SECURITY MANUAL, SECTION II-19 OR DOD 5200.1-R, INFORMATION SECURITY PROGRAM REGULATION, CHAPTER IX. FOR UNCLASSIFIED, LIMITED DOCUMENTS, DESTROY BY ANY METHOD THAT WILL PREVENT DISCLOSURE OF CONTENTS OR RECONSTRUCTION OF THE DOCUMENT.

LIMITED REPORTS

YOU MAY OBTAIN COPIES OF REPORTS LIMITED IN DISTRIBUTION ONLY IF YOU ARE WITHIN THE LIMITATION RANGE INDICATED UNLESS SPECIFICALLY AUTHORIZED BY THE CONTROLLING ORGANIZATION. TO REQUEST AUTHORIZATION, SEND DTIC-DDR A DTIC FORM 55, REQUEST FOR LIMITED DOCUMENT, AND INCLUDE ALL DESCRIPTIVE CATALOGING INFORMATION NECESSARY FOR ACCURATE IDENTIFICATION.

YOU MUST BE CERTIFIED WITH THE DEFENSE LOGISTICS SERVICES CENTER, BATTLE CREEK, MICHIGAN, TO OBTAIN COPIES OF REPORTS MARKED "EXPORT CONTROL" THROUGH DTIC.

COMPLAINTS

IF YOU RECEIVE DEMAND OR DIRECT RESPONSE BIBLIOGRAPHIES WHICH DO NOT MEET YOUR REQUIREMENTS CONTACT THE DEMAND PRODUCTS BRANCH, 202-274-6867 OR AUTOVON 284-6867. FOR CURRENT AWARENESS BIBLIOGRAPHIES CONTACT THE SPECIAL PRODUCTS AND TERMINOLOGY BRANCH, 202-274-7206 OR AUTOVON 284-7206. FOR BIBLIOGRAPHIES YOU HAVE ORDERED VIA A REMOTE ON-LINE TERMINAL CONTACT THE ON-LINE SUPPORT OFFICE 202-274-7206 OR AUTOVON 284-7709. PLEASE CITE THE SEARCH CONTROL NUMBER OF THE BIBLIOGRAPHY WHEN YOU CALL.

NOTICES

NOTICE

NOTICE- THIS TECHNICAL REPORT BIBLIOGRAPHY MAY CONTAIN ENTRIES CREATED BY DOD INFORMATION ANALYSIS CENTERS (IACS). DOCUMENTS REFERENCED BY THESE ENTRIES ARE NOT AVAILABLE FROM DTIC; HOWEVER, ADDITIONAL INFORMATION CONCERNING THE DOCUMENTS IS AVAILABLE FROM THE CONTRIBUTING IAC. CURRENT AWARENESS BIBLIOGRAPHIES (CAB) FROM THE IAC DATA BASES ARE NOT AVAILABLE FROM DTIC. IT IS SUGGESTED THAT USERS CONTACT THE APPROPRIATE IAC DIRECTLY FOR CURRENT AWARENESS SERVICES.

DTIC WILL PROVIDE MICROFICHE COPIES OF IAC ORIGINATED TECHNICAL REPORTS PUBLISHED AND PLACED ON FILE IN THE DTIC TECHNICAL REPORT COLLECTION AFTER JANUARY 1985 TO REGISTERED DTIC USERS AT THE STANDARD MICROFICHE PRICE. REQUIREMENTS FOR FIRST QUALITY HARDBOUND PAPER EDITIONS WILL CONTINUE TO BE FILLED BY THE IAC PRODUCING THE REPORT, OR IN SOME CASES BY NTIS, AT THE IAC COST RECOVERY PRICE.

IAC CREATED ENTRIES ARE RECOGNIZABLE BY THE IAC ACRONYM AND UNIQUE AD NUMBER ASSIGNMENT. THE FOLLOWING IAC ACRONYMS AND AD NUMBER ASSIGNMENTS IDENTIFY THE CONTRIBUTING IAC:

MCIC AD NUMBERS 175000-189999 AND D100000-D199999
METALS AND CERAMICS INFORMATION CENTER
BATTELLE -- COLUMBUS LABORATORIES
505 KING AVENUE COLUMBUS, OH 43201
TELEPHONE 614-424-5000

MMCIAC AD NUMBERS D200000-D299999
METAL MATRIX COMPOSITES INFORMATION ANALYSIS CENTER
KAMAN TEMPO
816 STATE STREET SANTA BARBARA, CA 93102
TELEPHONE 805-963-6455

NTIAC AD NUMBERS 1900000-199999 AND D300000-D399999
NONDESTRUCTIVE TESTING INFORMATION ANALYSIS CENTER
SOUTHWEST RESEARCH INSTITUTE
6220 CULEBRA ROAD P.O. DRAWER 28510 SAN ANTONIO, TX 78284
TELEPHONE 512-684-5111, EXT 2362

PLASTEC AD NUMBERS D400000-D499999
PLASTECS TECHNICAL EVALUATION CENTER
ARMY ARMAMENTS, MATERIALS, AND CHEMICALS COMMAND
BUILDING 351 NORTH DOVER, NJ 07801
TELEPHONE 201-724-4222 OR AUTOVON 880-4222

GACIAC AD NUMBERS D500000-D599999
GUIDANCE AND CONTROL INFORMATION ANALYSIS CENTER
ITT RESEARCH INSTITUTE
10 WEST 35TH STREET CHICAGO, IL 60616
TELEPHONE 312-567-4345

CPIA AD NUMBERS D600000-D699999
CHEMICAL PROPULSION INFORMATION AGENCY CENTER
THE JOHN HOPKINS UNIVERSITY
JOHN HOPKINS ROAD LAUREL, MD 20707
TELEPHONE 301-992-7300

SURVIAC AD NUMBERS D700000-D799999
SURVIVABILITY/VUNERABILITY INFORMATION ANALYSIS CENTER
BOOZ, ALLEN AND HAMILTON
AFWAL/FIESD/SURVIAC WRIGHT-PATTERSON AFB, OH 45433
TELEPHONE 513-255-4840/3956 OR AUTOVON 785-4840/3956

MTIAC AD NUMBERS D800000-D899999
MANUFACTURING TECHNOLOGY INFORMATION ANALYSIS CENTER
CASE AND COMPANY, INC.
MTIAC OPERATIONS 10 WEST 35TH STREET CHICAGO, IL 60616
TELEPHONE 312-567-4733

ADDITIONAL IACS ARE BEING ESTABLISHED WHICH WILL BE ADDED TO THIS LIST OF DOD IACS SHARING CITATION ENTRIES. THE SUBJECT AREAS TO BE COVERED BY THESE IACS ARE: HIGH TEMPERATURE MATERIALS - MECHANICAL, THERMOPHYSICAL AND ELECTRONIC PROPERTIES; CHEMICAL WARFARE/CHEMICAL BIOLOGICAL DEFENSE; CORROSION PREVENTION AND CONTROL.

OTHER DEPARTMENT OF DEFENSE INFORMATION ANALYSIS CENTERS MANAGED BY DTIC

LISTED BELOW ARE FOUR OTHER DOD INFORMATION ANALYSIS CENTERS WHICH ARE MANAGED BY DTIC. THE DOD INFORMATION ANALYSIS CENTERS ARE RESPONSIBLE FOR COLLECTING, STORING, REVIEWING, EVALUATING, SYNTHESIZING, REPACKAGING, AND DISSEMINATING AUTHORITATIVE SCIENTIFIC AND TECHNICAL INFORMATION IN FORMATS MOST USEFUL TO THE SCIENTISTS, ENGINEERS, AND TECHNICIANS THEY SUPPORT. THE CENTERS ARE HIGHLY SELECTIVE WITH REGARD TO THE QUALITY OF INFORMATION THEY DISSEMINATE. FROM THE LITERATURE ACQUIRED, THEY EXTRACT AND EVALUATE INFORMATION PERTINENT TO THE NEEDS OF THEIR USERS. THIS EVALUATED INFORMATION IS SYNTHESIZED AND PACKAGED IN A VARIETY OF USEFUL FORMATS. THE CENTERS GENERALLY OFFER THE FOLLOWING CATEGORIES OF PRODUCTS/SERVICES: TECHNICAL INQUIRY SERVICE; BIBLIOGRAPHIC INQUIRY SERVICE; SCIENTIFIC AND ENGINEERING REFERENCE WORKS; STATE-OF-THE-ART REPORTS; CRITICAL REVIEWS AND TECHNOLOGY ASSESSMENTS; AND CURRENT AWARENESS BULLETINS. FOR ADDITIONAL INFORMATION ON SPECIFIC CENTERS' SERVICES, YOU ARE ENCOURAGED TO WRITE OR CALL THE CENTER OF INTEREST TO YOU. FOR GENERAL INFORMATION CONCERNING THE CENTERS, YOU MAY CALL THE PROGRAM MANAGER FOR INFORMATION ANALYSIS CENTERS AT AUTOVON 304-6260 OR COMMERCIAL 202-274-6260.

RELIABILITY ANALYSIS CENTER (RAC)
ROME AIR DEVELOPMENT CENTER
ATTN: RADC/RAC
GRIFFISS AFB, NY 13441
315-330-4151 OR AUTOVON 587-4151

INFRARED INFORMATION & ANALYSIS CENTER (IRIA)
ENVIRONMENTAL RESEARCH INSTITUTE OF MICHIGAN
P.O. BOX 8618
ANN ARBOR, MI 48107
313-994-1200 EXT 214

DATA ANALYSIS CENTER FOR SOFTWARE (DACS)
IIT RESEARCH INSTITUTE
TURIN ROAD NORTH, P.O. BOX 180
ROME, NY 134 40
315-336-0937
AUTOVON 587-3395

*THERMOPHYSICAL & ELECTRONIC PROPERTIES ANALYSIS CENTER (TEPIAC)
PURDUE UNIVERSITY
2595 YEAGER ROAD
WEST LAFAYETTE, IN 47906
317-494-6300
317-463-1581

* NOTE: THE TEPIAC WILL BE REPLACED BY A NEW CENTER CALLED THE HIGH TEMPERATURE MATERIALS INFORMATION ANALYSIS CENTER (HTMIAC) IN FY 86.

SEARCH CONTROL NUMBER 058100

SEARCH STRATEGY

THE TERMS BELOW WERE SEARCHED BY THE COMPUTER. ASTERISK TERMS REPRESENT WEIGHTED RETRIEVAL TERMS. TRUNCATED RETRIEVAL TERMS INDICATE THAT ALL TERMS WITH THE DEPICTED ROOT HAVE BEEN SEARCHED. COORDINATE SEARCHES ARE PORTRAYED AS SEARCH TERMS LISTED ON VARIOUS LEVELS. EXCLUDED RETRIEVAL TERMS ARE DISPLAYED UNDER AN EXCLUDE LISTING.

FIRST LEVEL SEARCH TERMS

ARTIFICIAL INTELLIGENCE	
EXPERT	(TRUNCATED)
EXPERT	(TRUNCATED)
KNOWLEDGE BASE	(TRUNCATED)
KNOWLEDGE ENGINEERING	(TRUNCATED)

SECOND LEVEL SEARCH TERMS

ASSEMBLY LANGUAGES	
AUTOMATIC PROGRAMMING	
COMPILERS	
COMPUTER PROGRAM	(TRUNCATED)
COMPUTER PROGRAM DOCUMENTATION	
COMPUTER PROGRAMMING	
COMPUTER PROGRAMS	
CONTROL SEQUENCES	
DEBUGGING(COMPUTERS)	
EXECUTIVE ROUTINES	
FIELDS(COMPUTER PROGRAMS)	
FIRMWARE	
FORTRAN	
HIGH LEVEL LANGUAGES	
MACHINE CODING	
MACROPROGRAMMING	
MICROPROGRAMMING	
PROGRAMMING LANGUAGES	
SIMULATION LANGUAGES	
SOFTWARE TEST	(TRUNCATED)
SUBROUTINES	

THIRD LEVEL SEARCH TERMS

- COMPUTER PROGRAM RELIABILITY
- COMPUTER PROGRAM VERIFICATION
- DEBUGGING(COMPUTERS)
- EVALUATION
- TEST AND EVALUATION
- TEST SETS

TITLE INDEX

AD-P003 035 5/1 9/2

DUKE UNIV DURHAM NC

(U) Knowledge Acquisition and Evaluation within Expert Systems,

JAN 84 7P

PERSONAL AUTHORS: Loveland, D. W. ;

CONTRACT NO. AFOSR-81-0221

UNCLASSIFIED REPORT

SUPPLEMENTARY NOTE: This article is from 'Proceedings of the Army Conference on Application of Artificial Intelligence to Battlefield Information Management Held at White Oak, Maryland on April 20, 21, and 22, 1983,' AD-A139 685, p207-213.

ABSTRACT: (U) Knowledge acquisition and evaluation are essential to maintaining the expertise of expert systems. This paper summarizes some of the major efforts to date in knowledge acquisition and indicates work being undertaken in the just emerging but crucial area of knowledge evaluation. (Author)

DESCRIPTORS: (U) *Systems engineering, *Data acquisition, *Computer programs, *Computer program verification, Validation, Construction, State of the Art

IDENTIFIERS: (U) Component Reports, *Expert systems, *Knowledge acquisition

AD-P003 035

AD-B078 293 12/1 9/2

JET PROPULSION LAB PASADENA CA

(U) Intelligence Algorithm Methodology I.

DESCRIPTIVE NOTE: Final rept.,

AUG 83 132P

PERSONAL AUTHORS: Gillis, J. W. ;Griesel, M. A. ;Kuo, T. J. ; Radbill, J. R. ;

REPORT NO. JPL-D-183

CONTRACT NO. NAS-7-918

UNCLASSIFIED REPORT

ABSTRACT: (U) This is one in a series of algorithm analysis reports on work performed at the Jet Propulsion Laboratory for the US Army Intelligence Center and School covering selected algorithms in existing Intelligence and Electronic Warfare (IEW) systems. It focuses on developing a methodology to characterize and catalogue IEW algorithms, and comparison of different algorithms is discussed. The report does not provide a detailed manual for analysis and cataloging, although a specific method is used as an example. Rather, it provides the structure within an algorithm analysis effort can be designed. Specific schema are suggested for descriptive parameters, including a characterization by mathematical field and military application. The characteristics of evaluative parameters (accuracy, timing, memory requirements, operating environment) are also discussed, and analysis parameters and techniques (robustness, proof of correctness) are examined.

DESCRIPTORS: (U) *Algorithms, Numerical methods and procedures, Identification, Parameters, Classification, Test and evaluation, Comparison, Information retrieval, Military applications, Artificial intelligence, Computer programs, Information processing, Statistical decision theory, Probability, Interpolation, Approximation(Mathematics)

IDENTIFIERS: (U) Algorithm analysis, Intelligence algorithm methodology, Algorithm report, Algorithm comparison, S/L change 8519

AD-B078 293

AD-A175 211 5/8

DENVER RESEARCH INST CO SOCIAL SYSTEMS RESEARCH AND EVALUATION DIV

(U) Artificial Intelligence Technology for the Maintainer's Associate.

DESCRIPTIVE NOTE: Final rept. Oct 83-Dec 85,

DEC 86 83P

PERSONAL AUTHORS: Richardson, J. J. ;Anselme, Cindy J. ; Harmon, Kenneth R. ;Keller, Robert A. ;Moul, Bonita L. ;

CONTRACT NO. F33615-82-C-0013

PROJECT NO. 1121

TASK NO. 09

MONITOR: AFHRL
TR-86-31

UNCLASSIFIED REPORT

ABSTRACT: (U) Shortcomings in the ability of the armed services to maintain sophisticated equipment are recognized. Trends in technological sophistication, personnel resources, and warfare scenarios are expected to aggravate the situation. In view of policies regarding integrated diagnostics and a reduced reliance on paper based documentation, the concept of an interactive, portable, computer based maintainer's associate is proposed. This effort developed the technology for the Maintainer's Associate based on artificial intelligence techniques and demonstrate a prototype system in the field. The prototype Maintainer's Associate was developed for troubleshooting the F-111 6883 intermediate level avionics test station. The project included conceptual design, development and delivery software programming, delivery hardware prototyping, knowledge base development, field demonstration, and analysis of lessons learned. Several important issues were examined: hybrid diagnostics, knowledge engineering costs, user interfaces, and the integration of training and job aiding. The term 'hybrid diagnostics' refers to the utilization of multiple sources of knowledge in the development of maintenance expert systems, in particular dependency modeling and heuristic expertise of field technicians. In

the area of dependency modeling, one source of knowledge identified for the prototype was the test program set of the automatic test station. This information provided the specific measurement values and locations necessary for making measurements during troubleshooting. Knowledge engineering costs were controlled through use of these test programs sets and the development of 'glass box' enter which permitted knowledge base modifications during program operation.

DESCRIPTORS: (U) *MAN COMPUTER INTERFACE, *MAINTENANCE MANAGEMENT, MAINTENANCE PERSONNEL, PROTOTYPES, AVIONICS, MODELS, ARTIFICIAL INTELLIGENCE, MODIFICATION, COSTS, DIAGNOSIS(GENERAL), AUTOMATIC, TEST FACILITIES, COMPUTER PROGRAMMING, DEMONSTRATIONS, INTEGRATION, MAINTENANCE, MEASUREMENT, TEST SETS, SCENARIOS, WARFARE, HUMAN RESOURCES, USER NEEDS, INTERFACES

IDENTIFIERS: (U) Maintainers associate, PE62205F, WUAFHRL11210915

AD-A174 567 9/2 6/4

MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB

(U) ARLO: Another Representation Language Offer.

DESCRIPTIVE NOTE: Technical rept.,

OCT 86 98P

PERSONAL AUTHORS: Haase,Kenneth W. , Jr;

REPORT NO. AI-TR-901

CONTRACT NO. N00014-85-K-0124

UNCLASSIFIED REPORT

ABSTRACT: (U) This paper describes ARLO, a representation language language loosely modelled after Greiner and Lenat's RLL-1. ARLO is a structure-based representation language for describing structure-based representation languages, including itself. A given representation language is specified in ARLO by a collection of structures describing how its descriptions are interpreted, defaulted, and verified. This high level description is compiled into LISP code and ARLO structures whose interpretation fulfills the specified semantics of the language. In addition, ARLO itself --- as a language for expressing and compiling partial and complete language specifications --- is described and interpreted in the same manner as the language it describes and implements. This self description can be extended or modified to expand or alter the expressive power of ARLO's initial configuration. Languages which describe themselves --- like ARLO --- provide powerful mediums for systems which perform automatic self-modification, optimization, debugging, or documentation. AI systems implemented in such a self-descriptive language can reflect on their own capabilities, applying general problem solving and learning strategies to enlarge or correct them.

DESCRIPTORS: (U) *PROGRAMMING LANGUAGES, *ARTIFICIAL INTELLIGENCE, SELF OPERATION, REFLECTION, SEMANTICS, STRUCTURAL PROPERTIES, MODIFICATION, AUTOMATIC, OPTIMIZATION, DEBUGGING(COMPUTERS)

IDENTIFIERS: (U) *Representation languages, ARLO

AD-A174 567

AD-A174 567 CONTINUED

language, RLL(Representation Language Languages), Knowledge representation languages, Meta representation, Artificial intelligence languages, Self descriptive languages, Learning machines, Explanation systems

AD-A174 567

AD-A173 993 9/2

MCLEAN RESEARCH CENTER INC VA

(U) Expert System for Software Quality Assurance.

DESCRIPTIVE NOTE: Final rept. Apr-Oct 86,

NOV 86 189P

PERSONAL AUTHORS: Baum,William E. ;Podell,Judith ; Romstedt,G. N. ;

CONTRACT NO. DAAK70-84-D-0052

UNCLASSIFIED REPORT

ABSTRACT: (U) This report describes the development of an expert system for software quality assurance(SQA). The expert system was designed to facilitate the process of tailoring statements of work by capturing the knowledge of SQA engineers. This task was undertaken in order to alleviate the problems of staff turnover and inexperience and to ensure that the standards and requirement of an adequate SQA Program are enforced, thereby improving the level of performance of the BRDEC SQA mission.

DESCRIPTORS: (U) *COMPUTER PROGRAM RELIABILITY, *QUALITY ASSURANCE, FEASIBILITY STUDIES, SYSTEMS ENGINEERING, STANDARDS, REQUIREMENTS, SPECIFICATIONS

IDENTIFIERS: (U) *Export Systems

AD-A173 993

AD-A167 910 9/2

JET PROPULSION LAB PASADENA CA

(U) Intelligent Computer Assisted Instruction (ICAI): Formative Evaluation of Two Systems.

DESCRIPTIVE NOTE: Final rept. Apr 84-Aug 85.

MAR 86 356P

CONTRACT NO. NAS7-918

PROJECT NO. 2Q263743A794

MONITOR: ARI
RN-86-29

UNCLASSIFIED REPORT

ABSTRACT: (U) This report reviews major components of an 18 month evaluation of Intelligent Computer Assisted Instruction (ICAI), and emerging field of Artificial Intelligence that draws on computer technologies and cognitive science in an attempt to build more powerful instructional programs. The primary goals of this effort were to develop an increased understanding of the state of the art of ICAI for the purposes of: (a) identifying strategies to enhance the general usefulness of ICAI technology for Army training problems, and (b) developing concepts for efficiently and effectively managing military ICAI projects. The approach taken to accomplish these goals was to intensively examine two selected ICAI systems using a formative evaluation methodology. The two systems selected were: (a) PROUST, a system designed by Soloway and Johnson for analyzing bugs in novice programmers' PASCAL programs, using a top-down approach which attempts to infer the intentions and plans of the programmer, and (b) WEST, a system designed by Burton and Brown to teach basic mathematics and strategic thinking skills, based on the premise that students can learn from their mistakes or 'bugs'.

DESCRIPTORS: (U) *COMPUTER AIDED INSTRUCTION, ARMY TRAINING, COGNITION, ARTIFICIAL INTELLIGENCE, INTELLIGENCE, MATHEMATICS, STUDENTS, MILITARY APPLICATIONS, DEBUGGING(COMPUTERS), COMPUTER PROGRAMMING

IDENTIFIERS: (U) WU102, AS794, PE63743A

AD-A167 910

AD-A157 505 9/2

YALE UNIV NEW HAVEN CT DEPT OF COMPUTER SCIENCE

(U) Micro-PROUST.

DESCRIPTIVE NOTE: Research rept.,

JUN 85 123P

PERSONAL AUTHORS: Johnson,W. L. ;Soloway,E. ;

REPORT NO. YALEU/CSD/RR-402

CONTRACT NO. N00014-82-K-0714

UNCLASSIFIED REPORT

ABSTRACT: (U) PROUST(Program Understander for Students) is a knowledge-based system that finds nonsyntactic bugs in Pascal programs written by novice programmers. When students compile a program successfully, PROUST is automatically invoked to analyze it. PROUST reports any bugs that are in the program to the student. PROUST is a 15,000 LISP program and runs on a VAX. Micro-PROUST is a program meant to capture the essence of PROUST. Micro-PROUST is a 1500 line LISP program and runs on an IBM PC (with 512K). This document presents the inner workings of Micro-PROUST. Its intent is to enable those who so are inclined to see at a nuts and bolts level hcw a system like PROUST actually works. Additional keywords: intelligent tutoring systems; student modelling; automatic program debugging.

DESCRIPTORS: (U) *COMPUTER PROGRAMS, *DEBUGGING(COMPUTERS), PROGRAMMERS, STUDENTS

IDENTIFIERS: (U) PROUST(Program Understander for Students), Pascal programming language, Knowledge based systems, WUNR154492

AD-A157 505

AD-A153 379 9/2 20/8

SYSTEM PLANNING CORP ARLINGTON VA

(U) Evidential Reasoning in Expert Systems for Image Analysis.

DESCRIPTIVE NOTE: Final technical rept. Jun-Dec 84,

FEB 85 98P

PERSONAL AUTHORS: Thompson,T. R. ;

CONTRACT NO. DACA72-84-C-0006

MONITOR: ETL
0381

UNCLASSIFIED REPORT

ABSTRACT: (U) This report documents efforts to understand approaches to evidential reasoning of use in application of expert-system or knowledge-based-system techniques to image analysis (IA). There is growing evidence that these techniques offer significant improvements in image analysis, particularly in the coordinated application of specialized algorithms. This effort has four principal goals: (1) to clarify the basic issues in evidential reasoning (ER), (2) to provide a common framework for analysis, (3) to structure the ER process for major expert-system tasks in image analysis, and (4) to identify promising directions for further research. This research was carried out in three major segments. The first segment structured the evidential reasoning problem in a formal paradigm robust enough to be of practical use in design and construction of expert systems. It then formulated six important theoretical approaches in a parallel fashion in order to identify key assumptions, similarities, and differences. The second segment applied each of the ER approaches to three important tasks for expert systems in the domain of image analysis. This segment concluded with an assessment of the strengths and weaknesses of each approach. The third segment addressed promising directions for further research. It reviewed current results and identified important questions bearing on successful application of expert-system technology to image analysis.

DESCRIPTORS: (U) *COMPUTER PROGRAMS, *IMAGE PROCESSING, *REASONING, ALGORITHMS, CONSTRUCTION, TEST AND EVALUATION,

AD-A153 379

AD-A153 379 CONTINUED

THEORY

IDENTIFIERS: (U) *Expert systems

AD-A148 125 5/7 9/2

MARYLAND UNIV COLLEGE PARK DEPT OF COMPUTER SCIENCE

(U) A Small Contribution to Editing with a Syntax Directed Editor,

MAY 84 9P

PERSONAL AUTHORS: Zelkowitz,M. V. ;

CONTRACT NO. F49620-83-K-0018

PROJECT NO. 2304

TASK NO. A2

MONITOR: AFOSR
TR-84-0935

UNCLASSIFIED REPORT

SUPPLEMENTARY NOTE: Pub. in Software Engineering Noters, v9 n3, SIGPLAN Notices, v19 n5 May 84.

Reprint: A Small Contribution to Editing with a Syntax Directed Editor.

DESCRIPTORS: (U) *Syntax, *Editing, *Artificial intelligence, Error detection codes, Debugging(Computers), Translations, Text processing, Grammars, Semantics, Programming languages, Computer programming, Reprints

IDENTIFIERS: (U) *Syntax directed editor, SUPPORT(Still Unnamed Production Programming Oriented Research Tool), PE61102F, WUAFOSR2304A2

AD-A143 459 9/2 5/2

MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR COMPUTER SCIENCE

(U) Laboratory for Computer Science Progress Report 19, 1 July 1981-30 June 1982.

DESCRIPTIVE NOTE: Annual progress rept.,

MAY 84 287P

PERSONAL AUTHORS: Dertouzos, M. L. ;

REPORT NO. MIT/LCS-PR-19

CONTRACT NO. N00014-75-C-0661

UNCLASSIFIED REPORT

SUPPLEMENTARY NOTE: See also Progress rept. no. 18, AD-A127 586.

ABSTRACT: (U) This report summarizes the research performed at the MIT Laboratory for Computer Science from July 1, 1981 through June 30, 1982. The Contents Include: Computer Systems and Communications; Computer Systems Structures; Educational Computing Group; Functional Languages and Architecture Group; Information Mechanics; Message Passing Semantics; Office Automation; Programming Methodology; Programming Technology; Real Time Systems; Systematic Program Development.

DESCRIPTORS: (U) *Computer architecture, *Computer communications, *Computer programming, Computers, Minicomputers, Multiprocessors, Methodology, Semantics, Real time, Network flows, Network analysis(Management), Debugging(Computers), Man computer interface, Programming languages, Information processing, Artificial intelligence, Data bases, Data management, Reports

IDENTIFIERS: (U) Office automations, Computer science

AD-A143 459

AD-A135 503 9/2

STANFORD UNIV CA DEPT OF COMPUTER SCIENCE

(U) Simplification by Cooperating Decision Procedures,

APR 78 21P

PERSONAL AUTHORS: Nelson, G. ; Oppen, D. C. ;

REPORT NO. STAN-CS-78-652, AIM-311

CONTRACT NO. MDA903-76-C-0206, NSF-MCS76-000327

UNCLASSIFIED REPORT

SUPPLEMENTARY NOTE: Presented at the ACM Symposium on Principles of Programming Languages (5th), 1978.

ABSTRACT: (U) This report describes a simplifier for use in program manipulation and verification. The simplifier finds a normal form for any expression over the language consisting of individual variables, the usual boolean connectives, equality, the conditional function cond (denoting if-then-else), the numerals, the arithmetic functions and predicates +, - and <or=, the LISP constants, functions and predicates nil, car, cdr, cons and atom, the functions store and select for storing into and selecting from arrays, and uninterpreted function symbols. Individual variables range over the union of the reals, the set of arrays, LISP list structure and the booleans true and false. The simplifier is complete; that is, it simplifies every valid formula to true. Thus it is also a decision procedure for the quantifier-free theory of reals, arrays and list structure under the above functions and predicates. The organization of the simplifier is based on a method for combining decision procedures for several theories into a single decision procedure for a theory combining the original theories. More precisely, given a set S of functions and predicates over a fixed domain, a satisfiability program for S is a program which determines the satisfiability of conjunctions of literals (signed atomic formulas) whose predicate and function symbols are in S. We give a general procedure for combining satisfiability programs for sets S and T into a single satisfiability program for S u T, given certain conditions on S and T. The simplifier described in this paper is currently used in the Stanford Pascal Verifier.

AD-A135 503

AD-A135 503 CONTINUED

DESCRIPTORS: (U) *Computer program verification, Computer logic, Simplification, Model theory, Decision making, Computations, Functions(Mathematics), Symbols, Symbolic programming, Arrays, Algorithms, Artificial intelligence, Computer programs

IDENTIFIERS: (U) *Simplifier, Data structures

AD-A134 699 6/4 9/2 5/8

KESTREL INST PALO ALTO CA

(U) Report on a Knowledge-Based Software Assistant.

DESCRIPTIVE NOTE: Final technical rept. Jun 82-Jun 83,

AUG 83 78P

PERSONAL AUTHORS: Green,C. ;Luckham,D. ;Balzer,R. ; Cheatham,T. ;Rich,C. ;

CONTRACT NO. F30602-81-C-0206

PROJECT NO. 5581

TASK NO. 19

MONITOR: RADC
TR-83-195

UNCLASSIFIED REPORT

ABSTRACT: (U) This report presents a knowledge-based, life-cycle paradigm for the development, evolution, and maintenance of large software projects. To resolve current software development and maintenance problems, this paradigm introduces a fundamental change in the software life cycle - maintenance and evolution occur by modifying the specifications and then rederiving the implementation rather than attempting to directly modify the optimized implementation. It also describes a knowledge-based software assistant (KBSA) that provides for the capture of, and reasoning about, software activities to support this new paradigm. This KBSA will provide a corporate memory of the development history and act throughout the life cycle as a knowledgeable software assistant to the human involved (e.g., the developers, maintainers, project managers, and end-users). The report presents descriptions for several of the facets (areas of expertise) of the software assistant including requirements, specification validation, performance analysis, development, testing, documentations, and project management. This report also presents a plan for the development of the KBSA, along with a description of the necessary supporting technology.

DESCRIPTORS: (U) *Artificial intelligence, *Systems engineering, *Computer aided design, *Computer programs,

AD-A134 699 CONTINUED

Automatic programming, Maintainability, Life cycles, Life expectancy(Service life), Specifications, Requirements, Decision making, Problem solving, Computer program reliability, Computer program documentation, Computer program verification

IDENTIFIERS: (U) *Expert systems, *KBSA(Knowledge Based Software Assistant), Computer assisted paradigms, Paradigms, PE61102F, PE62702F, WURADC558119P5

IAC NO. MT-000875

IAC DOCUMENT TYPE: MTIAC - HARD COPY --

IAC SUBJECT TERMS: T--(U)*Software Development, *Knowledge Based Systems, Reliability, Productivity, Expert Systems, Artificial Intelligence, /Code T.;

AD-A134 699

AD-A133 692 9/5 9/2

RUTGERS - THE STATE UNIV NEW BRUNSWICK NJ DEPT OF COMPUTER SCIENCE

(U) A Knowledge Based Approach to VLSI CAD.

DESCRIPTIVE NOTE: Interim technical rept.,

SEP 83 21P

PERSONAL AUTHORS: Steinberg,Louis I. ;Mitchell,Tom M. ;

REPORT NO. 4-25664

CONTRACT NO. N00014-81-K-0394, ARPA Order-4147

UNCLASSIFIED REPORT

ABSTRACT: (U) Artificial Intelligence (AI) techniques offer one possible avenue toward new CAD tools to handle the complexities of VLSI. This paper summarizes the experience of the Rutgers AI/VLSI group in exploring applications of AI to VLSI design over the past few years. In particular, it summarizes our experience in developing REDESIGN, a knowledge-based system for providing interactive aid in the functional redesign of digital circuits. Given a desired change to the function of a circuit, REDESIGN combines rule-based knowledge of design tactics with its ability to analyze signal propagation through circuits, in order to (1) help the user focus on an appropriate portion of the circuit to redesign, (2) suggest local redesign alternatives, and (3) determine side effects of possible redesigns. We also summarize our more recent research toward constructing a knowledge-based system for VLSI design and a system for chip debugging, both based on extending the techniques used by the REDESIGN system. (Author)

DESCRIPTORS: (U) *Integrated circuits, *Computer aided design, *Artificial intelligence, Logic circuits, Chips(Electronics), Gates(Circuits), Debugging(Computers), Interactions, Computer programs, User needs, Digital systems, Approach, Propagation, Side reactions, Tools, Circuits, Focusing, Signals

IDENTIFIERS: (U) Knowledge based systems, 6574 memories, VLSI(Very Large Scale Intergration), 74166 shift resistors, Redesign computer program, Slice indices, 74175 latches, Character slices, Vexed computer program

AD-A133 692

AD-A133 447   9/2

YALE UNIV NEW HAVEN CT DEPT OF COMPUTER SCIENCE

(U) PROUST: Knowledge-Based Program Understanding.

DESCRIPTIVE NOTE: Technical rept.,

AUG 83   35P

PERSONAL AUTHORS: Johnson,W. Lewis ;Soloway,Elliot ;

REPORT NO. YALEU/DCS/RR-285

CONTRACT NO. N00014-82-K-0714

UNCLASSIFIED REPORT

ABSTRACT: (U) This paper describes a program called PROUST which does on-line analysis and understanding of Pascal programs written by novice programmers. PROUST takes as input a program and a non-algorithm description of the program requirements, and finds the most likely mapping between the requirements and the code. This mapping is in essence a reconstruction of the design and implementation steps that the programmer went through in writing the program. A knowledge base of programming plans and strategies, together with common bugs associated with them, is used in constructing this mapping. Bugs are discovered in the process of relating plans to the code; PROUST can therefore give deep explanations of program bugs by relating the buggy code to its underlying intentions. (Author)

DESCRIPTORS: (U) *Computer programs, *Debugging(Computers), *Computer programming, Planning, Artificial intelligence, Requirements, Mapping, Goal programming, Decomposition, On line systems

IDENTIFIERS: (U) PROUST computer program, Expert systems, WUNR154492

AD-A133 447

AD-A133 080   9/2   14/2   5/10   15/3

INTEGRATED SCIENCES CORP SANTA MONICA CA

(U) Evaluating the Effectiveness of Military Decision Support Systems. Theoretical Foundations, Expert System Design, and Experimental Plan.

DESCRIPTIVE NOTE: Research note,

SEP 82   66P

PERSONAL AUTHORS: Leal,Antonio ;

REPORT NO. ISC-345-3

CONTRACT NO. MDA903-81-C-0449

PROJECT NO. 2Q161102B74F

MONITOR: ARI
RN-83-18

UNCLASSIFIED REPORT

ABSTRACT: (U) The main objective of this program is to construct a flexible testbed for the evaluation of the effectiveness of computer-based expert systems in military training and planning. The technical approach consists of simulating the characteristics of expert systems in a game-like environment. Such characteristics include friendly system user interaction, system explanations of rationale about decision recommendations, an ability to make relevant suggestions and comments about situation assessments and about plans proposed by the user, and the use of high-level strategic concepts and terminology. The required software for such a program includes a game environment simulator called the Scenario Generator, a simulated expert system for the game called the Expert Aid, an Optimality Algoirthm for computing the best decisions in any situation, and an Evaluation Module for recording execution histories and performance parameters. The expert system will monitor the progress of the game and can be interrogated as the user sees fit. A facility will also be provided for evaluating the user's performance under different modes of consultation with the expert system.

DESCRIPTORS: (U) *Computer applications, *Computerized simulation, *Test and evaluation, *Decision making,

AD-A133 080

AD-A133 080 CONTINUED

*Military applications, Operational effectiveness, War games, Scenarios, Generators, Military planning, Military training, Game theory, Computer programs, Test beds

IDENTIFIERS: (U) Expert systems, PE61102A, AS74F

AD-A129 153 9/2 5/8

ADVANCED INFORMATION AND DECISION SYSTEMS MOUNTAIN VIEW CA

(U) The Intelligent Program Editor: A Knowledge Based System for Supporting Program and Documentation Maintenance.

DESCRIPTIVE NOTE: Technical rept.,

MAR 83 9P

PERSONAL AUTHORS: Shapior,Daniel G. ;McCune,Brian P. ;

CONTRACT NO. F49620-81-C-0067

PROJECT NO. 2304

TASK NO. A2

MONITOR: AFOSR
TR-83-0488

UNCLASSIFIED REPORT

ABSTRACT: (U) This paper presents work in progress towards a program development and maintenance aid called the Intelligent Program Editor (IPE), which applies artificial intelligence techniques to the task of manipulating and analyzing programs. The IPE is a knowledge based tool: it gains its power by explicitly representing textual, syntactic, and many of the semantic (meaning related) and pragmatic (application oriented) structures in programs. To demonstrate this approach, the authors implemented a subset of this knowledge base, and a search mechanism called the Program Reference Language (PRL), which is able to locate portions of programs based on a description provided by a user. This work is an applied research effort. It was motivated by issues discovered during a study of software maintenance problems in the Air Force, and is intended to be moved into application within seven years.

DESCRIPTORS: (U) *Computer programs, *Computer programming, *Editing, *Artificial intelligence, Data bases, Systems analysis, Text processing, Syntax, Semantics, Subroutines, Programming languages, Computer program documentation, Data management, Computer program verification, Models, Computer program reliability,

AD-A129 153 CONTINUED

Programmers, Man computer interface, User needs, Maintenance

IDENTIFIERS: (U) PE61102F, WUAFOSR2304A2

AD-A128 631 9/2 9/5

MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB

(U) A Simple Model of Circuit Design.

DESCRIPTIVE NOTE: Technical rept.,

MAY 80 68P

PERSONAL AUTHORS: Roylance, Gerald Lafael ;

REPORT NO. AI-TR-703

CONTRACT NO. N00014-80-C-0505, N00014-80-C-0622

UNCLASSIFIED REPORT

ABSTRACT: (U) A Simple analog circuit designer has been implemented as a rule based system. The system can design voltage followers, Miller integrators, and bootstrap ramp generators from functional descriptions of what these circuits do. While the designers works in a simple domain where all components are ideal, it demonstrates the abilities of skilled designers. While the domain is electronics, the design ideas are useful in many other engineering domains, such as mechanical engineering, chemical engineering, and numerical programming. Most circuit design systems are given the circuit schematic and use arithmetic constraints to select components values. This circuit designer is different because it designs the schematic. The designer uses a unidirectional CONTROL relation to find the schematic. The circuit designs are built around this relation; it restricts the search space, assigns purposes to components, and finds design bugs.

DESCRIPTORS: (U) *Computer aided design, *Models, *Circuits, *Analog systems, Automation, Artificial intelligence, Ramps, Generators, Schematic diagrams, Debugging(Computers), Feedback, Value engineering

IDENTIFIERS: (U) Expert systems, Circuit design, KCL(Kirchoff's Current Law)

AD-A127 793 9/2 14/2

OHIO STATE UNIV COLUMBUS DEPT OF COMPUTER AND INFORMATION SCIENCE

(U) Theoretical Foundations of Software Technology.

DESCRIPTIVE NOTE: Final scientific rept. 1 Jul 79-30 Sep 82,

FEB 83 133P

PERSONAL AUTHORS: Chandrasekaran,B. ;White,Lee J. ; Buttelmann,H. W. ;

CONTRACT NO. F49620-79-C-0152

PROJECT NO. 2304

TASK NO. A2

MONITOR: AFOSR
TR-83-0333

UNCLASSIFIED REPORT

SUPPLEMENTARY NOTE: Continuation of Grants AFOSR-77-3416 and AFOSR-75-2811.

ABSTRACT: (U) This is the final scientific report of research performed under the contract in various aspects of software technology. The research efforts can be categorized under three topics: computer program testing, knowledge-based systems for program construction, and theory of translator generation. In the first category researchers describe a number of research results relating to various aspects of domain testing strategy and integration testing of modules. In the second category, researchers decribe a program called LLULL, which understands programming problems stated in natural language in the domain of checking accounts, and produces PASCAL programs for them. In addition, researchers describe several projects in knowledge organization and problem solving. In the last category, researchers describe a research effort that focussed on obtaining theoretical results on the complexity of translator generation from one language to another.

DESCRIPTORS: (U) *Computer programs, *Test and evaluation, *Translators, Natural language, Problem

AD-A127 793

AD-A127 793 CONTINUED

solving, Diagnosis(General), Reasoning, Modules(Electronics), Integration, Computer programming, Artificial intelligence

IDENTIFIERS: (U) Knowledge representation, PE61102F, LPN-OSURF-761640/711991, WUAFOSR2304A2

AD-A127 793

AD-A126 413 9/2 6/4

CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST

(U) KBS: An Artificial Intelligence Approach to Flexible Simulation.

SEP 82 36P

PERSONAL AUTHORS: Reddy,Y. V. ;Fox,Mark S. ;

REPORT NO. CMU-RI-TR-82-1

UNCLASSIFIED REPORT

ABSTRACT: (U) This report describes KBS, a Knowledge-Based simulationsystem. The report describes the use of SRL, an AI-based knowledge representation system for modeling (e.g., factory organizations), and its interpretation of discrete simulations. KBS provides facilities for interactive model creation and alteration, simulation monitoring and control, graphics display, and selective instrumentation. It also allows the user to define and simulate a system at different levels of abstraction, and to check the completeness and consistency of a model, hence reducing model debugging time. (Author)

DESCRIPTORS: (U) *Computerized simulation, *Artificial intelligence, Computer architecture, Programming languages, Libraries, Circuit boards, Monitoring, Display systems, Interactive graphics, Instrumentation, Debugging(Computers)

IDENTIFIERS: (U) *Knowledge representation, KBS(Knowledge Based Simulation)

IAC NO. MT-000987

IAC DOCUMENT TYPE: MTIAC - HARD COPY --

IAC SUBJECT TERMS: T--(U)Artificial Intelligence, Knowledge Based Systems, Simulation, Programming Languages, /Code T.;

AD-A126 413

AD-A126 228 9/2

AIR FORCE HUMAN RESOURCES LAB BROOKS AFB TX

(U) PRISM: A General Purpose Programming System.

DESCRIPTIVE NOTE: Final technical paper,

MAR 83 17P

PERSONAL AUTHORS: Rogers,Charles R. ;O'Hara,Steven A. ;

REPORT NO. AFHRL-TP-82-44

PROJECT NO. 6323

TASK NO. 04

UNCLASSIFIED REPORT

ABSTRACT: (U) This paper describes the development, uses, and features of the general purpose programming system PRISM, which is the foundation for future program development by the Computer Programming Branch and is available to all personnel within the Air Force Human Resources Laboratory (AFHRL). PRISM was designed to meet the need for an efficient and reliable programming tool that could be used like a high-order programming language but still provide the operating system interface and hardware controls of assembly language. It has special features that make it an especially powerful tool for new software development. These features were derived from an extensive analysis of coding sequences in existing library programs, interactions between library programs, and the identification of common programming procedures. PRISM was specifically designed for the development of general purpose programs by the Technical Services Division of AFHRL within the Computer Programming Branch; however, it is also an effective and efficient tool for applications programmers. (Author)

DESCRIPTORS: (U) *Programming languages, Computer programs, Machine coding, Interactions, Libraries, Artificial intelligence, Computations, Algorithms, Debugging(Computers), Quality control, Maintenance, Subroutines, Computer files, Access

IDENTIFIERS: (U) Prism programming language, Univac 1100 computers, PE62703F, WUAFHRL63230423

AD-A126 228

AD-A125 647 9/2

NAVAL POSTGRADUATE SCHOOL MONTEREY CA

(U) Development of a Concurrent Tree Search Program.

DESCRIPTIVE NOTE: Master's thesis,

OCT 82 168P

PERSONAL AUTHORS: Powley,Curt Nelson ;

UNCLASSIFIED REPORT

ABSTRACT: (U) Search, especially tree search, is fundamental to the field of artificial intelligence. Even with good heuristic functions, the time it takes on a single processor to solve progressively more difficult tree search problems grows exponentially and quickly becomes constraining. It seems reasonable that the use of concurrency should significantly improve the speed of a tree search. After discussing concurrent programming issues as background, this thesis outlines some high-level approaches to concurrent tree search. Development of each high-level approach includes development of required operating system interfaces. With the warning that choosing the best approach requires empirical evaluation, a concurrent tree search algorithm for the eight-puzzle is presented. (Author)

DESCRIPTORS: (U) *Information retrieval, *Searching, *Computer programming, *Parallel processing, Algorithms, Artificial intelligence, Abstracts, Message processing, Exponential functions, Charts, Diagrams, Trees, Test and evaluation, Theses, Flow charting

IDENTIFIERS: (U) *Tree information retrieval*, *Tree search*, Decision trees

AD-A125 647

AD-A121 494 9/2

RAND CORP SANTA MONICA CA

(U) The ROSS Language Manual.

DESCRIPTIVE NOTE: Interim rept.,

SEP 82 58P

PERSONAL AUTHORS: McArthur,David ;Klahr,Philip ;

REPORT NO. RAND/N-1854-AF

CONTRACT NO. F49620-82-C-0018

UNCLASSIFIED REPORT

ABSTRACT: (U) This Note summarizes the commands of the ROSS language. ROSS is an object-oriented programming language currently being developed at Rand. The goal of ROSS is to provide a programming environment in which users can conveniently design, test and change large knowledge-based simulations of complex mechanisms. Object-oriented programming languages, and ROSS in particular, enforce a 'message-passing' style of programming in which the system to be modeled is represented as a set of actors and their behaviors (rules for actor interaction). This style is especially suited to simulation, since the mechanism or process to be simulated may have a part-whole decomposition that maps naturally onto actors. The first section of this Note gives an overall view of the language and the philosophy behind object-oriented programming. The next eleven sections give detailed descriptions of the basic commands or behaviors of the language. The final two sections give advice on how to write English-like code in ROSS and how to optimize code, once debugged. (Author)

DESCRIPTORS: (U) *Simulation languages, *Programming manuals, *Artificial intelligence, Message processing, Computerized simulation, Battles, Coding, English language, User needs, Optimization, Debugging(Computers)

IDENTIFIERS: (U) Object oriented programming language, Ross programming language

AD-A121 494

AD-A120 806 9/4 6/4

STANFORD UNIV CA DEPT OF COMPUTER SCIENCE

(U) An Approach to Verifying Completeness and Consistency in a Rule-Based Expert System.

DESCRIPTIVE NOTE: Technical rept.,

AUG 82 26P

PERSONAL AUTHORS: Suwa,Motoi ;Scott,A. Carlisle ; Shortliffe,Edward H. ;

REPORT NO. STAN-CS-82-922

CONTRACT NO. N00014-81-K-0004, NSF-MCS79-03735

UNCLASSIFIED REPORT

SUPPLEMENTARY NOTE: Sponsored in part by Grants PHS-RR-00785, PHS-LM-03395 and PHS-LM-00048.

ABSTRACT: (U) We describe a program for verifying that a set of rules in an expert system comprehensively spans the knowledge of a specialized domain. The program has been devised and tested within the context of the ONCOCIN System, a rule-based consultant for clinical oncology. The stylized format of ONCOCIN's rules has allowed the automatic detection of a number of common errors as the knowledge base has been developed. This capability suggests a general mechanism for correcting many problems with knowledge base completeness and consistency before they can cause performance errors.

DESCRIPTORS: (U) *Information systems, *Systems engineering, *Man computer interface, *Information transfer, User needs, Editing, Debugging(Computers), Problem solving, Error analysis, Embedding, Computer logic

IDENTIFIERS: (U) Knowledge base, ONCOCIN system

AD-A120 806

AD-A120 319 9/2 9/4

ADVANCED INFORMATION AND DECISION SYSTEMS MOUNTAIN VIEW CA

(U) Design of an Intelligent Program Editor.

DESCRIPTIVE NOTE: Final technical rept. 1 Jan-31 Jul 82,

SEP 82 108P

PERSONAL AUTHORS: Shapiro,Daniel G. ;McCune,Brian P. ; Wilson,Gerald A. ;

REPORT NO. AI/DS-TR-3023-1

CONTRACT NO. N00014-82-C-0119

UNCLASSIFIED REPORT

ABSTRACT: (U) This report discusses results of a project to develop a functional design for and assess the feasibility of an intelligent program editor for ADA and other programming languages. The editor will support program development and maintenance activities by providing advanced techniques for searching through programs, manipulating programs, analyzing programs for potential errors and good style, and maintaining structured documentation. These techniques are based on knowledge-based systems technology from the field of artificial intelligence. Feasibility of the program editor is demonstrated by a functional design and an initial implementation of the multiple knowledge bases representing a small program and a search (query) mechanism that uses them. The use of such an editor implies significant benefits for programmer productivity, program reliability, and life-cycle costs. (Author)

DESCRIPTORS: (U) *Artificial intelligence, *Computer programs, *Editing, Programming languages, Error analysis, Searching, Computer program documentation, Systems analysis, Semantics, Debugging(Computers), Maintenance, Human factors engineering, Man computer interface, Input output processing, Models, Data management, Data bases, Syntax, Computers, Operational effectiveness, Cost effectiveness

AD-A120 319

AD-A120 248 12/1

DUKE UNIV DURHAM NC DEPT OF CHEMISTRY

(U) Search Algorithms and Their Implementation.

DESCRIPTIVE NOTE: Annual rept. 1 Jul 81-30 Jun 82,

AUG 82 21P

PERSONAL AUTHORS: Loveland,D. W. ;

CONTRACT NO. AFOSR-81-0221

PROJECT NO. 2304

TASK NO. A2

MONITOR: AFOSR
TR-82-0878

UNCLASSIFIED REPORT

ABSTRACT: (U) Research that has resulted in completed papers involved (1) optimality of search procedures (decision trees) in binary testing; (2) a study of signature table representation for evaluation functions and methods for dynamically improving function accuracy; (3) pruning minimax trees that have been adapted to incorporate moves determined by chance; and (4) the search problem in automated program construction. Preliminary results have been obtained in research on (a) optimizing limited resource is to guide otherwise random search; (b) studying search strategies in two-person games when information is partly concealed; and (c) limiting search in debugging rule sets in one type of expert knowledge system. Other investigations are in progress. (Author)

DESCRIPTORS: (U) *Algorithms, *Searching, Problem solving, Decision making, Optimization, Strategy, Debugging(Computers), Minimax technique, Trees, Accuracy

IDENTIFIERS: (U) Expert systems, PE61102F, WUAFOSR2304A2

AD-A120 248

AD-A116 787 5/8 9/2 6/4

MASSACHUSETTS INST OF TECH CAMBRIDGE DEPT OF MECHANICAL ENGINEERING

(U) Computer Simulated Visual and Tactile Feedback as an Aid to Manipulator and Vehicle Control,

MAY 81 138P

PERSONAL AUTHORS: Winey,Calvin McCoy , III ;

CONTRACT NO. N00014-77-C-0256

UNCLASSIFIED REPORT

ABSTRACT: (U) A computer graphic simulation of a seven degree-of-freedom slave manipulator controlled by an actual master was developed. An electronically coupled E-2 manipulator had previously been interfaced to a PDP-11/34 by K. Tani, allowing the computer to sense and control each degree of freedom independently. The simulated manipulator was capable of moving an arbitrary shaped object and sensing a force in an arbitrary direction with no actual object or force existing. The simulated manipulator could also be attached to a simulated vehicle capable of motion with six degrees-of-freedom. The vehicle simulation is currently being used in conjunction with dynamic simulations developed by H. Kazerooni to test different types of dynamic controllers for submarines. Shadows, multiple views and proximity indicators were evaluated to determine their effectiveness in giving depth information. The results indicated that these aids are useful. Subjects felt that shadows gave the best perception of the environment, but found isometric views easist to use on the tasks performed. This type of simulation appears to be realistic and adaptable to a multitude of applications.

DESCRIPTORS: (U) *Man machine systems, *Man computer interface, *Computerized simulation, *Computer graphics, *Artificial intelligence, *Manipulators, Vehicles, Degrees of freedom, Feedback, Operators(Personnel), Shadows, Depth indicators, Submarine simulators, Teleoperators, Test and evaluation, Computer programs, Control systems, Interfaces, Proximity devices, Touch, Visual perception

IDENTIFIERS: (U) Robots, PDP 11/34 computer, WUNR196152

AD-A116 787

AD-A113 494 9/2

MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB

(U) Seeing What Your Programs are Doing,

FEB 82 41P

PERSONAL AUTHORS: Lieberman,Henry ;

REPORT NO. AI-M-656

CONTRACT NO. N00014-75-C-0522, N00014-80-C-0505

UNCLASSIFIED REPORT

ABSTRACT: (U) An important skill in programming is being able to visualize the operation of procedures, both for constructing programs and debugging them. TINKER is a programming environment for Lisp that enables the programmer to 'see what the program is doing' while the program is being constructed, by displaying the result of each step in the program on representative examples. To help the reader visualize the operation of Tinker itself. (Author)

DESCRIPTORS: (U) *Computer program verification, *Visual perception, *Computer graphics, *Screens(Displays), Artificial intelligence, Interactions, Debugging(Computers), Test and evaluation

IDENTIFIERS: (U) TINKER programming environment, Lisp programming language, Example based programming, Program testing

AD A113 494

AD-A110 224 9/2

SYSTEMS CONTROL INC PALO ALTO CA COMPUTER SCIENCE DEPT

(U) Codification of Program Synthesis Knowledge for Concurrent Programs - Year II.

DESCRIPTIVE NOTE: Final rept. 1 Jul 79-30 Sep 81,

SEP 81 16P

PERSONAL AUTHORS: Chapiro,Daniel ;

REPORT NO. SCI-ICS-L-81-1

CONTRACT NO. F49620-79-C-0137

PROJECT NO. 2304

TASK NO. A2

MONITOR: AFOSR
TR-81-0895

UNCLASSIFIED REPORT

ABSTRACT: (U) This report is the final report of our research project on the codification of concurrent programming knowledge. The general goal of research in this area is to codify programming knowledge and to create programming systems that employ this knowledge to assist in various programing activities including specification, synthesis, modification, debugging, and maintenance. The aim is to produce knowledge-based design tools to help with problems in this area. This paper primarily raises some questions that must be addressed in a study of a more focused area, namely that of generation of concurrent microcode. We first introduce a basic parallelism operator. The intent is to refine parallel programs specified using this operator into microcode. We discuss briefly how the hardware architecture affects the level of parallelism exploited in the microcode. Then we discuss issues in the automatic generation of compact yet fast microcode. Some advantages of microcode programming by refinement of high-level specifications are brought up, namely exploiting high-level parallelism, and assurance of correctness of the resulting code. The refinement paradigm requires intermediate level constructs and search for efficient implementations, which are discussed. An example is devised to see if macroparallelism in the

AD-A110 224

AD A110 224 CONTINUED

high level specification is carried over in the microcode.

DESCRIPTORS (U) *Artificial intelligence, *Computer programming, *Parallel processing, *Computer aided design, Data bases, Methodology, Debugging(Computers), Maintainability, Dual mode, Synthesis, Microprocessors, Computer architecture

IDENTIFIERS (U) *Concurrent programming, Programming environments, PE61102F, WUAFOSR2304A2

AD A110 224

AD-A110 030 9/2

MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB

(U) Inspection Methods in Programming.

DESCRIPTIVE NOTE: Technical rept.,

JUN 81 289P

PERSONAL AUTHORS: Rich,Charles ;

REPORT NO. AI-TR-604

CONTRACT NO. N00014-75-C-0643, N00014-80-C-0505

UNCLASSIFIED REPORT

SUPPLEMENTARY NOTE: Sponsored in part by Grant NSF-MCS79-12179.

ABSTRACT: (U) The work reported here lies in the area of overlap between artificial intelligence and software engineering. As research in artificial intelligence, it is a step towards a model of problem solving in the domain of programming. In particular, this work focuses on the routine aspects of programming which involve the application of previous experience with similar programs. I call this programming by inspection. Programming is viewed here as a kind of engineering activity. Analysis and synthesis by inspection are a prominent part of expert problem solving in many other engineering disciplines, such as electrical and mechanical engineering. The notion of inspection methods in programming developed in this work is motivated by similar notions in other areas of engineering. This report concentrates on the knowledge base of the programmer's apprentice, which is in the form of a taxonomy of commonly used algorithms and data structures. To the extent that a programmer is able to construct and manipulate programs in terms of the forms in such a taxonomy, he may relieve himself of many details and generally raise the conceptual level of his interaction with the system, as compared with present day programming environments. Also, since it is practical to expend a great deal of effort pre-analyzing the entries in a library, the difficulty of verifying the correctness of programs constructed this way is correspondingly reduced. The feasibility of this approach is demonstrated by the

AD-A110 030

CONTINUED

... library of common techniques for ... symbolic data.

DESCRIPTORS: (U) *Computer programming, *Computer program verification, *Computer aided diagnosis, Artificial intelligence, Decision making, Problem solving, Methodology, Algorithms, Taxonomy, Inspection, Synthesis, Man computer interface, Network flows

AD-A110 030

AD-A107 328 9/2 12/1

STANFORD UNIV CA DEPT OF COMPUTER SCIENCE

(U) Deductive Synthesis of the Unification Algorithm,

JUN 81 52P

PERSONAL AUTHORS: Manna,Zohar ;Waldinger,Richard ;

REPORT NO. STAN-CS-81-855

CONTRACT NO. N00014-76-C-0687, AFOSR-81-0014

MONITOR: AFOSR
TR-82-0492

UNCLASSIFIED REPORT

SUPPLEMENTARY NOTE: Prepared in cooperation with SRI International, Menlo Park, CA. Artificial Intelligence Center, Contract N00014-75-C-0816 and Weizmann Inst. of Science, Rehovoth (Israel). Sponsored in part by Grants NSF-MCS78-02591 and NSF-MCS79-09495.

ABSTRACT: (U) THE DEDUCTIVE APPROACH IS A FORMAL PROGRAM-CONSTRUCTION METHOD IN WHICH THE DERIVATION OF A PROGRAM FROM A GIVEN SPECIFICATION IS REGARDED AS A THEOREM-PROVING TASK. To construct a program whose output satisfies the conditions of the specification, we prove a theorem stating the existence of such an output. The proof is restricted to be sufficiently constructive so that a program computing the desired output can be extracted directly from the proof. The program we obtain is applicative and may consist of several mutually recursive procedures. The proof constitutes a demonstration of the correctness of this program. To exhibit the full power of the deductive approach, we apply it to a nontrivial example -- the synthesis of a unification algorithm. Unification is the process of finding a common instance of two expressions. Algorithms to perform unification have been central to many theorem-proving systems and some programming-language processors. The task of deriving a unification algorithm automatically is beyond the power of existing program-synthesis systems. In this paper, we use the deductive approach to derive an algorithm from a simple, high-level specification of the unification task. We will identify some of the capabilities required of a theorem-proving system to perform this derivation automatically. (Author)

AD-A107 328

AD-A107 328 CONTINUED

DESCRIPTORS (U) *Computer programming *Mathematical logic, *Algorithms Theorems Recursive functions, Debugging(Computers), Artificial intelligence Calculus

IDENTIFIERS (U) *Deductive approach *Theorem proving Correctness, Lambda calculus

AD A107 328

AD A105 661 9/2

SYSTEMS CONTROL INC PALO ALTO CA COMPUTER SCIENCE DEPT

(U) Research on Knowledge Based Programming and Algorithm Design

DESCRIPTIVE NOTE Final technical rept 27 Nov 78-31 Aug 81

AUG 81 121P

PERSONAL AUTHORS Green, Cordell ,

REPORT NO SCI ICS L 81 5

CONTRACT NO N00014 79 C 0127 N00014-80-C 0045

UNCLASSIFIED REPORT

ABSTRACT (U) The object of our research is the codification of programming knowledge and the creation of computer systems that incorporate this knowledge that assist in the various activities of programming. We have designed and implemented the CHI knowledge based programming system including the 'V' wide spectrum language for expressing both programming knowledge and program specifications. CHI has been used to synthesize several programs including parts of itself. We are extending the uses of the knowledge base to provide intelligent tools for environment to support not only program synthesis but program acquisition, modification, debugging and maintenance. Another aspect of our research is called Algorithm Design. This project emphasizes tools to assist in the more creative aspects of the creation of new algorithms. We have formalized a set of methods, primarily focused upon the incorporation of operations into generators that seem to be a very powerful set of tools in deriving good and difficult algorithms. We have implemented some of these methods in CHI and include a discussion of the derivations in this report

DESCRIPTORS (U) *Computer programming *Artificial intelligence *Computer aided design Algorithms, Data bases Debugging(Computers) Maintainability Dynamic programming Data management Methodology Computer program documentation Environments

IDENTIFIERS (U) *CHI computer program LPN ARPA Order 3687 LPN ARPA Order 3828

AD A105 661

AD-A105 515 9 2

CALIFORNIA UNIV BERKELEY ELECTRONICS RESEARCH LAB

(U) Embedding Expert Knowledge and Hypothetical Data Bases into a Data Base System

DESCRIPTIVE NOTE Memorandum rept

APR 80 17P

PERSONAL AUTHORS Stonebraker,M ;Keller,K

REPORT NO UCB ERL M80 15

CONTRACT NO N00039 76 C 0022, DAAG29 76 G 0245

UNCLASSIFIED REPORT

ABSTRACT (U) This paper is concerned with adding knowledge to a data base management system and suggests two appropriate mechanisms, namely hypothetical data bases (HDB s) and experts. Herein we indicate the need for HDB s and define the extensions that are needed to a data base system to support HDB s. In addition, we suggest that the notion of 'experts' is an appropriate way to add semantic knowledge to a data base system. Unlike most other proposals which extend an underlying data model to capture more meaning, our proposal does not require extensions to the schema. Moreover, the DBMS does not even have to know how an expert functions. In this paper we define an expert and indicate how it would be added to one existing data base system. (Author)

DESCRIPTORS (U) *Data bases, *Data management, *Artificial intelligence, Semantics, Computer programming, Computer architecture, Embedding, Methodology, Debugging(Computers), Information retrieval, Multivariate analysis

IDENTIFIERS (U) *Data base management systems, HDB(Hypothetical Data Bases)

AD-A105 515

AD-A102 157 9 2

MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB

(U) Abstraction, Inspection and Debugging in Programming.

DESCRIPTIVE NOTE Memorandum rept.,

JUN 81 33P

PERSONAL AUTHORS Rich,Charles ;Waters,Richard C. ;

REPORT NO AI M 634

CONTRACT NO N00014 80 C-0505, NSF-MCS79-12179

UNCLASSIFIED REPORT

ABSTRACT (U) We believe that software engineering has much to learn from other mature engineering disciplines, such as electrical engineering, and that the problem solving behaviors of engineers in different disciplines have many similarities. Three key ideas in current artificial intelligence theories of engineering problem solving are: Abstraction -- using a simplified view of the problem to guide the problem solving process. Inspection -- problem solving by recognizing the form ('plan') of a solution. Debugging -- incremental modification of an almost satisfactory solution to a more satisfactory one. These three techniques are typically used together in a paradigm which we call AID (for Abstraction, Inspection, Debugging): First an abstract model of the problem is constructed in which some important details are intentionally omitted. In this simplified view inspection methods are more likely to succeed, yielding the initial form of a solution. Further details of the problem are then added one at a time with corresponding incremental modifications to the solution. This paper states the goals and milestones of the remaining three years of a five year research project to study the fundamental principles underlying the design and construction of large software systems and to demonstrate the feasibility of a computer aided design tool for this purpose, called the programmers apprentice. (Author)

DESCRIPTORS (U) *Computer programming, *Methodology, *Computer aided design, *Artificial intelligence, Problem solving, Debugging(Computers), Automation, Systems

AD-A102 157

AD-A102 157 CONTINUED

analysis, Goal programming, Fault tree analysis

IDENTIFIERS: (U) AID(Abstraction Inspection Debugging)

AD-A101 101 9/2 17/2 15/7 5/1
17/9

NAVAL POSTGRADUATE SCHOOL MONTEREY CA

(U) Evaluation of the Artificial Intelligence Program STAMMER2 in the Tactical Situation Assessment Problem.

DESCRIPTIVE NOTE: Master's thesis,

MAR 81 90P

PERSONAL AUTHORS: Ferranti,John Peter , Jr;

UNCLASSIFIED REPORT

ABSTRACT: (U) STAMMER2 (System for Tactical Assessment of Multisource Messages, Even Radar) is an experimental program created as part of an investigation into methods of correlating information in the naval environment. This thesis is an exploration into the application of artificial intelligence to the tactical situation assessment problem and into various evaluation methodologies for STAMMER2. Included is an overview of one of these experiments, using the facilities of the Naval Postgraduate School Command, Control and Communications Laboratory and the Naval Ocean Systems Center, San Diego. (Author)

DESCRIPTORS: (U) *Artificial intelligence, *Tactical analysis, *Research management, Information processing, Correlation techniques, Naval operations, Threat evaluation, Radar scanning, Systems analysis, Message processing, Computer program documentation, Command and control systems, Decision making, Problem solving, Test methods, Multisensors, Data acquisition, Test and evaluation, Theses

IDENTIFIERS: (U) TSA(Tactical Situation Assessment), STAMMER2 computer program, STAMMER(System for Tactical Assessment of Multisource Messages Even Radar), Deductive reasoning

AD-A099 174 9/2 12/1 15/7

RAND CORP SANTA MONICA CA

(U) An Analysis of Proximity-Detection and Other Algorithms in the ROSS Simulator.

DESCRIPTIVE NOTE: Interim rept.,

MAR 81 48P

PERSONAL AUTHORS: Faught, William S. ;Klahr, Philip ;

REPORT NO. RAND/N-1587-AF

CONTRACT NO. F49620-77-C-0023

UNCLASSIFIED REPORT

ABSTRACT: (U) This report summarizes the mechanisms by which the ROSS simulator computes interactions (collisions and proximities) between objects. ROSS simulates an air penetration scenario and is being developed to research techniques for improving large-scale simulation. The basic algorithm is analyzed in detail to determine its feasibility in the context of large numbers of objects, and to determine where improvements in speed can occur. (Author)

DESCRIPTORS: (U) *Computerized simulation, *Probability density functions, *Aerial warfare, *Computer program documentation, War games, Artificial intelligence, User needs, Parameters, Numerical analysis, Test and evaluation, Computer program verification, Target detection, Algorithms

IDENTIFIERS: (U) ROSS computer programs, Proximity detection

AD-A099 174

*AD-A097 037 9/2 17/2

NAVAL OCEAN SYSTEMS CENTER SAN DIEGO CA

(U) Command, Control and Communications(C3) Systems Model and Measures of Effectiveness (MOEs).

DESCRIPTIVE NOTE: Technical rept. Jul-Sep 80,

OCT 80 30P

PERSONAL AUTHORS: Harmon, S. Y. ;Brandenburg, R. L. ;

REPORT NO. NOSC/TR-598

PROJECT NO. X0738-CC

TASK NO. X0738-CC

UNCLASSIFIED REPORT

ABSTRACT: (U) The logical structure of a new comprehensive C3 system model which is independent of national origin and tactical situation and which forms the basis for development of a computer simulation for analysis of C3 systems performance is introduced. Three classes of C3 MOEs which when taken together completely describe all the critical elements of C3 system's performance are also introduced. One of these MOE classes (ie, the MU class) includes most of the MOEs which have been proposed and utilized previously. In addition, two other classes are proposed which include a measure of the effects of information consistency (ie, the ALPHA-BETA class) as well as a completely new class of MOEs describing the knowledge differences between the elements of a force and between two opposing forces (ie, the DELTA-K class). Each of these classes of MOEs has both local and global interpretations which permit the evaluation of the component parts of a C3 system as well as of the performance of the C3 system as a whole. (Author)

DESCRIPTORS: (U) *Command and control systems, *Computerized simulation, Test and evaluation, Artificial intelligence, Tactical communications, Tactical data systems, Data transmission systems, Network flows, Nodes, Communications networks, Decision making, Computer programming, Data acquisition, Input output processing, Computer logic, Data bases

IDENTIFIERS: (U) *CE(Command Control and Communications),

AD-A097 037

AD-A097 037 CONTINUED

Robots, Distributed data processing, PE65858N

AD-A095 521 9/2

MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB

(U) A Session with TINKER: Interleaving Program Testing With Program Design.

DESCRIPTIVE NOTE: Memorandum rept.,

SEP 80 38P

PERSONAL AUTHORS: Lieberman, Henry ; Hewitt, Carl ;

REPORT NO. AI-M-577

CONTRACT NO. N00014-75-0643, N00014-75-C-0522

UNCLASSIFIED REPORT

ABSTRACT: (U) Tinker is an experimental interactive programming system which integrates program testing with program design. New procedures are created by working out the steps of the procedure in concrete situations. Tinker displays the results of each step as it is performed, and constructs a procedure for the general case from sample calculations. The user communicates with Tinker mostly by selecting operations from menus on an interactive graphic display rather than by typing commands. This paper presents a demonstration of our current implementation of Tinker. (Author)

DESCRIPTORS: (U) *Computer programming, *Man computer interface, *Computer program verification, *Computer program documentation, Interactive graphics, Interactions, Computer program reliability, Artificial intelligence, User needs, Instruction manuals, Computations, Dynamic programming

IDENTIFIERS: (U) *Interactive computer system, LISP programming language, TINKER computer program

AD-A093 186 9/2

MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB

(U) EMACS Manual for ITS Users

DESCRIPTIVE NOTE: Memorandum rept.

JUN 80 194P

PERSONAL AUTHORS: Stallman, Richard M.

REPORT NO: AI-M-554

CONTRACT NO: N00014-75-C-0643

UNCLASSIFIED REPORT

Availability: Document partially illegible

ABSTRACT: (U) This manual documents the use and simple customization of the display editor EMACS with the ITS operating system. The reader is not expected to be a programmer. Even simple customizations do not require programming skill, but the user who is not interested in customizing can ignore the scattered customization hints. This is primarily a reference manual, but can also be used as a primer. (Author)

DESCRIPTORS: (U) *Text processing, *Dynamic programming, *Display systems, *Data displays, *Computer program documentation, Instruction manuals, User needs, Control systems, Self operation, Artificial intelligence, Real time, Specifications, Debugging(Computers), Computer programming, Computer files

IDENTIFIERS: (U) ITS operating system, Emacs display editor

AD-A093 186

AD-A078 060 9/2

MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB

(U) Computer Aided Evolutionary Design for Software Engineering.

DESCRIPTIVE NOTE: Memorandum rept.,

JAN 79 23P

PERSONAL AUTHORS: Rich, Charles ;Shrobe, Howard E. ;Waters, Richard C. ;

REPORT NO: AI-M-506

CONTRACT NO: N00014-75-C-0643, N00014-75-C-0661

UNCLASSIFIED REPORT

ABSTRACT: (U) This memorandum reports on a partially implemented interactive computer aided-design tool for software engineering. A distinguishing characteristic of this project is its concern for the evolutionary character of software systems. It draws a distinction between algorithms and systems, centering its attention on support for the system designer. Although verification has played a large role in recent research, this perspective suggests that the complexity and evolutionary nature of software systems require a number of additional techniques. Managing of complexity is a fundamental issue in all engineering disciplines. The authors identify three major techniques used in mature engineering fields which seem applicable to the engineering of software systems: incremental modelling; multiple and almost hierachical decomposition; and analysis by inspection. Along these lines they have (1) Constructed a plan library to aid in analysis by inspection (the analysis of a program based on identifying standard algorithms and methods in it); (2) Identified a small set of plan building methods which can be used to decompose a softwre system into loosely coupled subsystems; (3) Developed the technique of temporal abstraction which makes it possible to model a program from a viewpoint which clearly separates the actions of generations and consumers of data, and (4) Developed a dependency-based reasoning system uniquely suited to incremental and evolutionary program analysis. These methods are substantially language independent and have been applied to programs

AD-A078 060

AD A078 060 CONTINUED

written in several commonly used languages

DESCRIPTORS: (U) *Computer aided design, *Computer programming, Artificial intelligence, Algorithms Decomposition, Computer program verification, Computer logic, Flow charting, Interactions, Automatic programming, Problem solving

IDENTIFIERS: (U) Structured programming

AD A068 838 6/4 9/4 9/2 5/7

MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB

(U) Progress in Artifical Intelligence 1978 Volume 1

DESCRIPTIVE NOTE: Technical rept.

79 456P

PERSONAL AUTHORS: Winston, Patrick H ;Brown, Richard H. ;

CONTRACT NO: N00014-75-C-0643 N00014-77-C-0389

UNCLASSIFIED REPORT

SUPPLEMENTARY NOTE: See also Volume 2, AD-A068 839.

ABSTRACT: (U) This two volume collection was assembled to introduce advanced topics in Artificial Intelligence and to characterize the MIT point of view. With this in mind, the selected contributions are meant to be representative of either the research area explored or the methodology employed. Some of the shorter selections appear in full in order to convey a feeling for the detail and precision required in implementing working programs. Usually, however, length considerations have forced considerable abridgment. This necessarily means that the sections often describe what can be done but not much about how. With one exception, all of the sections originally appeared as publications of the MIT Artificial Intelligence Laboratory. Volume I includes reports on the general topics of : Expert Problem Solving; Natural Language Understanding and Intelligent Computer Coaches; and Representation and Learning.

DESCRIPTORS: (U) *Artificial intelligence, *Problem solving, *Natural language, *Information processing, Computer applications, Learning machines, Man computer interface, Reasoning, Learning, Pattern recognition, Computer programming, Information transfer, Debugging(Computers)

AD-A052 952 9/2 14/2

MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB

(U) Qualitative Knowledge, Causal Reasoning, and the Localization of Failures.

DESCRIPTIVE NOTE: Doctoral thesis,

NOV 76 193P

PERSONAL AUTHORS: Brown, Allen Leon , Jr;

REPORT NO. AI-TR-362

CONTRACT NO. N00014-70-A-0362-0003

UNCLASSIFIED REPORT

ABSTRACT: (U) This report investigates some techniques appropriate to representing the knowledge necessary for understanding a class of electronic machines -- radio receivers. A computational performance model 'WATSON' is presented. WATSON's task is to isolate failures in radio receivers whose principles of operation have been appropriately described in the knowledge base (Author)

DESCRIPTORS: (U) *Computer aided diagnosis, Failure(Electronics), Radio receivers, Circuits, Algorithms, Test sets, Computer programming, Fault tree analysis, computer logic, Maintenance, Automation, Artificial intelligence, Theses

IDENTIFIERS: (U) Fault detection

AD-A052 952

AD-A052 726 9/2

TRW DEFENSE AND SPACE SYSTEMS GROUP REDONDO BEACH CALIF

(U) Automated Compiler Test Case Generation.

DESCRIPTIVE NOTE: Final technical rept. Apr 76-Dec 77,

FEB 78 111P

PERSONAL AUTHORS: Berning, Paul t. ;Anderson, Eric R. ;Belz, Frank C. ;

CONTRACT NO. F30602-76-C-0255

PROJECT NO. 5581

TASK NO. 12

MONITOR: RADC TR-78-30

UNCLASSIFIED REPORT

ABSTRACT: (U) This report discusses the overall design of a software tool for the automation of validation of compilers for conformance to the specification of the high-order programming language they process. Such compiler validation is currently tedious, time-consuming, error-prone, and not completely effective. The generation of test cases for compiler validation is here envisioned as a two-step process. The starting point for this implementation is the SEMANOL tool, which consists of a machine-representable exact specification of a high-order language that is used to check the consistency of the HOL specification and to directly execute programs written in that HOL. SEMANOL is used by the Analyzer to generate constraints to which compiler test cases must adhere. The Synthesizer then uses these constraints to generate the test cases via a tree-building process. Although a considerable degree of human intervention is still required in the development of compiler test programs when a tool such as the one designed under this effort is employed, a vast improvement in the process of compiler validation and verification expected to result.

DESCRIPTORS: (U) *Compilers, *Computer programs, Automation, Programming languages, Specifications, Computer program verification, Artificial intelligence, Error detection codes

AD-A052 726

AD-A052 726 *CONTINUED*

IDENTIFIERS: (U) Semanol programming language, PE62702F, WURADC55811218

AD-A052 440 9/2

STANFORD RESEARCH INST MENLO PARK CALIF

(U) QA4: A Procedural Calculus for Intuitive Reasoning.

DESCRIPTIVE NOTE: Technical note,

NOV 72 354P

PERSONAL AUTHORS: Rulifson,Johns F. ;Derksen,Jan A. ; Waldinger,Richard J. ;

REPORT NO. TN-73

CONTRACT NO. NASW-2086

UNCLASSIFIED REPORT

ABSTRACT: (U) This report presents a language, called QA4, designed to facilitate the construction of problem-solving systems used for robot planning, theorem proving, and automatic program synthesis and verification. QA4 integrates an omega-order logic language with canonical composition, associative retrieval, and pattern matching of expressions; process structure programming; goal-directed searching; and demons. Thus it provides many useful programming aids. More importantly, however, it provides a semantic framework for common sense reasoning about these problem domains. The interpreter for the language is extraordinarily general, and is therefore an adaptable tool for developing the specialized techniques of intuitive, symbolic reasoning used by the intelligent systems.

DESCRIPTORS: (U) *Computer logic, *Computer program verification, *Problem solving, Artificial intelligence, Programming languages, Pattern recognition, Heuristic methods, Control, Automation, Semantics

IDENTIFIERS: (U) LISP programming language, ALGOL programming language, LPN-SRI-8721

AD-A052 211 5/9 9/2 6/4 1/2

MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB

(U) Annotated Production Systems: A Model for Skill Acquisition.

FEB 77 25P

PERSONAL AUTHORS: Goldstein, Ira P. ;Grimson, Eric ;

REPORT NO. AI-M-407, LOGO-M-44

CONTRACT NO. N00014-75-C-0643, N61339-76-C-0046

UNCLASSIFIED REPORT

ABSTRACT: (U) APS provide a procedural model for skill acquisition by augmenting a production model of the skill with formal commentary describing plans, bugs, and interrelationships between various productions. This commentary supports processes of efficient interpretation, self-debugging and self-improvement. The theory of annotated productions is developed by analyzing the skill of attitude instrument flying. An annotated production interpreter has been written that executes skill models which control a flight simulator. (Author)

DESCRIPTORS: (U) *Flight simulators, *Computer aided instruction, Computer programs, Computerized simulation, Skills, Instrument flight, Artificial intelligence, Debugging(Computers), Attitude control systems, Flight maneuvers

AD-A052 211

AD-A050 135 9/2

CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE

(U) Carnegie-Mellon University Multi-Microprocessor Review.

DESCRIPTIVE NOTE: Interim rept.,

JUN 77 91P

PERSONAL AUTHORS: Fuller, S. H. ;Jones, A. K. ;Durham, I. ;

CONTRACT NO. F44620-73-C-0074, ARPA Order-2466

MONITOR: AFOSR TR-78-0115

UNCLASSIFIED REPORT

ABSTRACT: (U) The Cm* project has been in progress almost exactly two years. The present review is intended to critically examine our progress to date and evaluate our plans for the future of Cm*. This report contains a description of the measurement and evaluation studies conducted. However, it was desirable to evaluate CM* as early in its development cycle as possible to detect and correct flaws in our design and understanding of this experimental multiprocessor. The present state of Cm* is described and all of our current measurements tabulated so a diligent reader can study and evaluate for himself our recent results.

DESCRIPTORS: (U) *Microprocessors, *Multiprocessors, *Computer program documentation, Time, Problem solving, Partial differential equations, Sorting, Integer programming, Speech recognition, Artificial intelligence

IDENTIFIERS: (U) Evaluation, Algol programming language, Algol 68 programming language, PE61101E

AD-A050 135

AD-A046 703 9/2 12/1

STANFORD UNIV CALIF DEPT OF COMPUTER SCIENCE

(U) The Logic of Computer Programming.

DESCRIPTIVE NOTE: Technical rept.,

AUG 77 87P

PERSONAL AUTHORS: Manna,Zohar ;Waldinger,Richard ;

REPORT NO. STAN-CS-77-611, AIM-298

CONTRACT NO. N00014-75-C-0816, N00014-76-C-0687

UNCLASSIFIED REPORT

ABSTRACT: (U) Techniques derived from mathematical logic promise to provide an alternative to the conventional methodology for constructing, debugging, and optimizing computer programs. Ultimately, these techniques are intended to lead to the automation of many of the facets of the programming process. This paper provides a unified tutorial exposition of the logical techniques, illustrating each with examples. The strengths and limitations of each technique as a practical programming aid are assessed and attempts to implement these methods in experimental systems are discussed. (Author)

DESCRIPTORS: (U) *Computer logic, *Computer programming, Optimization, Algorithms, Mathematical logic, Artificial intelligence, Computer architecture, Computer program verification

IDENTIFIERS: (U) WUNR049378, WUNR049389

AD-A045 102 9/2

STANFORD UNIV CALIF DEPT OF COMPUTER SCIENCE

(U) Sail,

AUG 76 183P

PERSONAL AUTHORS: Reiser,John F. ;

REPORT NO. STAN-CS-76-574, AIM-289

CONTRACT NO. MDA903-76-C-0206

UNCLASSIFIED REPORT

ABSTRACT: (U) Sail is a high-level programming language for the PDP-10 computer. It includes an extended ALGOL 60 compiler and a companion set of execution-time routines. In addition to ALGOL, the language features: (1) flexible linking to hand-coded machine language algorithms, (2) complete access to the PDP-10 I/O facilities, (3) a complete system of compile-time arithmetic and logic as well as a flexible macro system, (4) a high-level debugger, (5) records and references, (6) sets and lists, (7) an associative data structure, (8) independent processes, (9) procedure variables, (10) user modifiable error handling, (11) backtracking, and (12) interrupt facilities. This manual describes the Sail language and the execution-time routines for the typical Sail user: a non-novice programmer with some knowledge of ALGOL. It lies somewhere between being a tutorial and a reference manual. (Author)

DESCRIPTORS: (U) *High level languages, *Computer programming, Associative processing, Algorithms, Compilers, Computer programs, Debugging(Computers), Instruction manuals, Artificial intelligence

IDENTIFIERS: (U) *Sail programming language, ALGOL, PDP-10 computers

DTIC REPORT BIBLIOGRAPHY SEARCH CONTROL NO 058100

AD-A044 231 9/2 9/4

STANFORD UNIV CALIF DEPT OF COMPUTER SCIENCE

(U) Recent Research in Computer Science

DESCRIPTIVE NOTE: Annual rept. Jul 76-Jun 77,

JUN 77 123P

PERSONAL AUTHORS: McCarthy, John ;Binford, Thomas ;Green, Cordell ;Luckham, David ;Manna, Zohar ;

REPORT NO. STAN-CS-77-624, AIM-301

CONTRACT NO. MDA903-76-C-0206, ARPA Order-2494

UNCLASSIFIED REPORT

ABSTRACT: (U) This report summarizes recent accomplishments in six related areas: basic AI research and formal reasoning, image understanding, mathematical theory of computation, program verification, natural language understanding, and knowledge based programming.

DESCRIPTORS: (U) *Computer programming, *Artificial intelligence, Reasoning, Computer program verification, Natural language, Programming languages, Photointerpretation, Pattern recognition, Image processing, Digital systems, Computer logic, Reports

AD-A044 231

AD-A042 516 9/2 6/4 12/2

STANFORD UNIV CALIF DEPT OF COMPUTER SCIENCE

(U) The Evolution of Programs: A System for Automatic Program Modification

DESCRIPTIVE NOTE: Technical rept.,

DEC 76 47P

PERSONAL AUTHORS: Dershowitz, Nachum ;Manna, Zohar ;

REPORT NO. STAN-CS-76-586, AIM-294

CONTRACT NO. MDA903-76-C-0206, AF-AFOSR-2909-76

UNCLASSIFIED REPORT

ABSTRACT: (U) An attempt is made to formulate techniques of program modification, whereby a program that achieves one result can be transformed into a new program that uses the same principles to achieve a different goal. For example, a program that uses the binary search paradigm to calculate the square root of a number may be modified to divide two numbers in a similar manner, or vice versa. Program debugging is considered as a special case of modification: if a program computes wrong results, it must be modified to achieve the intended results. The application of abstract program schemata to concrete problems is also viewed from the perspective of modification techniques. We have embedded this approach in a running implementation; our methods are illustrated with several examples that have been performed by it. [illegible]

DESCRIPTORS: [illegible] *Artificial intelligence, *Computer programming, Automatic programming, Computer programs, [illegible] Debugging (Computers), Mathematical programming

AD A042 507

STANFORD UNIV [illegible] ARTIFICIAL [illegible]

(U) Is Sometimes [illegible]
Intermittent A[illegible] [illegible] Programming

DESCRIPTIVE NOTE [illegible]

APR 77 [illegible]

PERSONAL AUTHORS [illegible]

REPORT NO STAN [illegible]

CONTRACT NO [illegible]

SUPPLEMENTARY NOTE [illegible]
AD A031 4[illegible] [illegible]
0266 and ARPA [illegible]

ABSTRACT (U) This paper [illegible] proving the [illegible] simultaneously [illegible] intermittent [illegible] program with [illegible] when control is [illegible] but that need not [illegible] introduced by [illegible] complement [illegible] intermittent [illegible] of examples of correctness and termination [illegible] of these proofs are markedly simpler than their conventional counterparts. On the other hand, [illegible] that a proof of correctness or termination by [illegible] conventional techniques can be rephrased directly as a proof using intermittent assertions. Finally, [illegible] the intermittent assertion method can be applied to prove the validity of program transformations and the correctness of continuously operating programs [illegible] [illegible] methods [illegible] Artificial intelligence [illegible] systems, Algorithms

DESCRIPTORS (U) *Computer program reliability, Computer programs, Computer logic, Artificial intelligence, Input output processing, Transformational grammars, Invariance Hypotheses

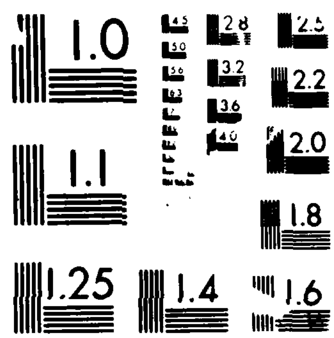
IDENTIFIERS (U) Computer program termination, WUNR049378, WUNR049389

AD A042 507

AD A038 245

AD-A187 034 FEASIBILITY STUDY OF GENERATING PLANS AND STRATEGIES FOR SOFTWARE TESTING. (U) SONEX ENTERPRISES INC FAIRFAX VA D J MATCHINSKI 13 OCT 87 DAAB07-87-C-F012 2/2

UNCLASSIFIED F/G 12/5 NL



MICROCOPY RESOLUTION TEST CHART

AD-A038 244 9/2

MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB

(U) Symbolic Evaluation Using Conceptual Respresentations for Programs with Side-Effects.

DESCRIPTIVE NOTE: Memorandum rept.,

DEC 76 44P

PERSONAL AUTHORS: Yonezawa, Akinori ;Hewitt, Carl ;

REPORT NO. AI-M-399

CONTRACT NO. N00014-75-C-0522

UNCLASSIFIED REPORT

ABSTRACT: (U) Symbolic evaluation is a process which abstractly evaluates a program on abstract data. A formalism based on conceptual representations is proposed as a specification language for programs with side-effects. Relations between algebraic specifications and specifications based on conceptual representations are discussed and limitations of the current algebraic specification techniques are pointed out. Symbolic evaluation is carried out with explicit use of a notion of situations. Uses of situational tags in assertions make it possible to state relations about properties of objects in different situations. The proposed formalism can deal with problems of side-effects which have been beyond the scope of Floyd-Hoare proof rules and give a solution to McCarthy's frame problem. (Author)

DESCRIPTORS: (U) *Symbolic programming, *Computer program verification, Programming languages, Artificial intelligence, Concept formation, Debugging(Computers), Queueing theory, Computations, Theory

AD-A038 244

AD-A037 925 9/2

RAND CORP SANTA MONICA CALIF

(U) A Computer and Its Man,

JUN 76 8P

PERSONAL AUTHORS: Shapiro, Norman ;

REPORT NO. P-5681

UNCLASSIFIED REPORT

ABSTRACT: (U) Having read a book (Computer Power and Human Reason by Joseph Weizenbaum, W. H. Freeman and Company, San Francisco, 1976), by the program's creator, it is no longer clear to me who the joke was on. Apparently, some people took the program seriously. The book's principal avowed purpose is to show that there are certain activities that computers (because they are computers) ought not to be made to engage in and derivatively that there are certain kinds of research that ought not to be done. In this endeavor, the author has, I think, substantially failed.

DESCRIPTORS: (U) *Computer applications, *Man computer interface, Artificial intelligence, Psychotherapy, Syntax, Speech recognition, Books, Reviews, Fortran, Compilers, Debugging(Computers), Natural language, Computers

IDENTIFIERS: (U) Eliza computer program

AD-A037 925

AD-A036 977   5/7   6/4

MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB

(U) Overview of a Linguistic Theory of Design.

DESCRIPTIVE NOTE: Memorandum rept.,

FEB 77   32P

PERSONAL AUTHORS: Miller,Mark L. ;Goldstein,Ira P. ;

REPORT NO. AI-M-383A, Logo-M-30A

CONTRACT NO. N00014-75-C-0643, MDA903-76-C-0108

UNCLASSIFIED REPORT

SUPPLEMENTARY NOTE: See also AD-A036 915.

ABSTRACT: (U) The SPADE theory uses linguistic formalisms to model the program planning and debugging processes. The theory has been applied to constructing a grammar-based editor in which programs are written in a structured fashion, designing an automatic programming system based on an Augmented Transition Network, and parsing protocols of programming episodes.

DESCRIPTORS: (U) *Artificial intelligence, *Linguistics, *Computational linguistics, Theory, Problem solving, Information processing, Psychology, Planning, Debugging(Computers), Computer programming, Taxonomy, Grammars, Syntax

IDENTIFIERS: (U) SPADE theory, Cognitive psychology, Structured programming

AD-A036 977

AD-A036 915   5/7   12/1   9/2

MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB

(U) Overview of a Linguistic Theory of Design.

DESCRIPTIVE NOTE: Memorandum rept.,

DEC 76   32P

PERSONAL AUTHORS: Miller,Mark L. ;Goldstein,Ira P. ;

REPORT NO. AI-M-383, Logo-M-30

CONTRACT NO. N00014-75-C-0643, NSF-C40708

UNCLASSIFIED REPORT

ABSTRACT: (U) SPADE is a theory of the design of computer programs in terms of complementary planning and debugging processes. An overview of the author's recent research on this theory is provided. SPADE borrows tools from computational linguistics -- grammars, augmented transition networks (ATN's), chart-based parsers -- to formalize planning and debugging. The theory has been applied to parsing protocols of programming episodes, constructing a grammar-based editor in which programs are written in a structured fashion. (Author)

DESCRIPTORS: (U) *Computational linguistics, *Mathematical programming, Problem solving, Artificial intelligence, Psychology, Information processing, Planning, Debugging(Computers), Computer programs

AD-A036 915

AD-A036 815 5/7 12/1 9/2

MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB

(U) Structured Planning and Debugging. A Linguistic Theory of Design,

DEC 76 87P

PERSONAL AUTHORS: Goldstein, Ira P. ;Miller, Mark L. ;

REPORT NO. AI-M-387, Logo-M-34

CONTRACT NO. N00014-75-C-0643, NSF-C-40708

UNCLASSIFIED REPORT

SUPPLEMENTARY NOTE: Revised version of Rept. nos. AI-Working Paper-125 and Logo Working Paper-55.

ABSTRACT: (U) A unified theory of planning and debugging is explored by designing a problem solving program called PATN. PATN uses an augmented transition network (ATN) to represent a broad range of planning techniques, including identification, decomposition, and reformulation. (The ATN is a simple yet powerful formalism which has been effectively utilized in computational linguistics). PATN's plans may manifest 'rational bugs', which result from heuristically justifiable but incorrect arc transitions in the planning ATN. This aspect of the theory is developed by designing a complementary debugging module called DAPR, which would diagnose and repair the errors in PATN's annotated plans. The investigation is incomplete: PATN has not yet been implemented. But sufficient detail is presented to provide a theoretical framework for reconceptualizing Sussman's HACKER research. Since a detailed study of planning and debugging techniques is a prerequisite for complete fulfillment of Dijkstra's objectives of program reliability, readability, portability, and so on, the theory is called, 'Structured Planning and Debugging', to emphasize its potential role in this enterprise.

DESCRIPTORS: (U) *Computational linguistics, *Mathematical programming, Problem solving, Debugging(Computers), Planning, Artificial intelligence, Networks, Computer programs

AD-A036 815

AD-A035 943 9/2 5/9

MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB

(U) Initial Report on a LISP Programmer's Apprentice.

DESCRIPTIVE NOTE: Technical rept.,

DEC 76 213P

PERSONAL AUTHORS: Rich, Charles ;Shrobe, Howard E. ;

REPORT NO. AI-TR-354

CONTRACT NO. N00014-75-C-0643, N00014-75-C-0522

UNCLASSIFIED REPORT

ABSTRACT: (U) The conceptual basis of the system lies in three forms of program description: (1) definition of structured data objects, their parts, properties, and relations between them, (2) input-output specification of the behavior of program segments (specs), and (3) a hierachical representation of the internal structure of programs (plans). The major theoretical work reported here is a representation for program plans which includes data flow, control flow, and also goal-subgoal prerequisite, and other dependency relationships between the segments of the program. (Author)

DESCRIPTORS: (U) *Computer programming, *Programmers, Computer programs, Computer program verification, Artificial intelligence, Debugging(Computers), Automatic, Specifications, Computer program documentation, Machine coding, Hierarchies

IDENTIFIERS: (U) *Programmers apprentice, Scenarios, Computer software

AD-A035 943

Data & Analysis Center for Software
RADC/COED
Griffiss AFB, NY 13441-5700

(315) 336-0937
Autovon: 587-3395
DDN: DACS @ RADC-Multics

The Data & Analysis Center for Software (DACS) is a Department of Defense Information Analysis Center operated for the Defense Logistics Agency (DLA) and Rome Air Development Center (RADC) by IIT Research Institute (IITRI).

DACS Bibliographic Search

Run: April 8, 1987

Search Strategy

[KEYWORD=TS*] and [KEYWORD=AI*]
or
[KEYWORD=TS*] and [KEYWORD=APEX*]

6 Record(s) Selected

1. Selected Citations

The select citations are listed in ascending order by Document Accession Number (DAN).

DAN	Citation
1461	WEGNER, PETER,EDITOR; "RESEARCH DIRECTIONS IN SOFTWARE TECHNOLOGY - INTRODUCTION AND OVERVIEW," In 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING. 0(0): May 1978. PP. 1-38. Also in RESEARCH DIRECTIONS IN SOFTWARE TECHNOLOGY. 0(0): Jan 1979. Avail. from MIT Press, 28 Carleton Street, Cambridge, MA 02142. Keywords: DATABASE MANAGEMENT SYSTEMS, ARCHITECTURE, AUTOMATIC PROGRAMMING, PERFORMANCE, PROGRAM SYNTHESIS, DEVELOPMENTAL METHODOLOGIES, NATURAL LANGUAGES, DISTRIBUTED PROCESSING, CONCURRENT PROGRAMMING, TESTING, SPECIFICATIONS, MANAGEMENT, SOFTWARE ENGINEERING, COMPLEXITY, EDUCATION

The organization of this book is explained, the changing technological environment is reviewed, and capsule descriptions of each paper (chapter) are included in this introduction. The first four chapters (Part I) consider the nature of the software problem and describe concepts and tools for managing large software systems. The remaining 16 chapters (Part II) describe and analyze specific research areas, divided into three subareas: (1) software methodology (managerial and technical issues); (2) computer system methodology (computers, languages, system programs); and (3) application methodology (broad application areas). Discussion items are included at the end of each Part or subarea which further explore specific research areas or offer alternative points of view.

1784 HOFFMAN, KARLA; WITZGALL, CHRISTOPH; "A LEXICAL SYNTHESIS APPROACH TO USER-ORIENTED INPUT SPECIFICATION," In 17TH ANNUAL TECHNICAL SYMPOSIUM,NBS/ACM. 0(0): Jun 1978. PP. 179-185. Avail. from ACM, Inc., 1133 Avenue OF Americas, New York, NY 10036.

Keywords: TEXT-PROCESSING APPLICATIONS, CODE READING, CODE VERIFICATION, NATURAL LANGUAGES

This paper presents a general and highly flexible "lexical synthesis" approach to the lexical decoding problem based on systematic string recognition rather than delimiting rules. It has successfully been implemented in an operating general-purpose lexical synthesis package ULEX.

3939 CHAPMAN, DAVID; "A PROGRAM TESTING ASSISTANT," COMMUNICATIONS OF THE ACM. 25(9): Sep 1982. PP. 625-634. Contract/Grant No. MCS-7912179, Sponsored by NATIONAL SCIENCE FOUNDATION. Contract/Grant No. N00014-80-C-0505, Sponsored by Office of Naval Research, Quincy Street, Arlington, VA 22217.

Keywords: TRANSFORMATION, LISP, SOFTWARE TOOLS, PROGRAMMING AIDS, DEBUGGING, SOFTWARE ENGINEERING TOOLS AND TECHNIQUES, ARTIFICIAL INTELLIGENCE, AUTOMATED TESTING

This article describes the design and implementation of a program testing assistant for the MIT Artificial Intelligence Laboratory's LISP Machine. The program testing assistant aids a programmer in the definition, execution, and modification of test cases during incremental program development. A brief overview of the testing assistant is first presented. Next, an example scenario of the assistant's use is given, with commentary and additional explanation of topics introduced in the overview, and implementation issues and techniques are examined. Finally, the testing assistant is related to other research.

4615 DEAN, JEFFREY S.; MCCUNE, BRIAN P.; ADVANCED TOOLS FOR SOFTWARE MAINTENANCE. Technical Report TR-3006-1. Contract/Grant No. F30602-80-C-0176, Sponsored by Rome Air Development Center, Griffiss AFB, Rome, NY 13441-5700. Avail. from National Technical Information Service 5285 Port Royal Rd, Springfield, VA 22161.

Keywords: UNIX, PRODUCTIVITY, VERIFICATION, EDITORS, PROGRAM MAINTENANCE, MODIFICATION PROCEDURES, COMMAND,CONTROL, AND COMMUNICATIONS, TESTING, AUTOMATED DOCUMENTATION, KNOWLEDGE BASED SYSTEMS, ARTIFICIAL INTELLIGENCE, INTERLISP, MAINTENANCE TOOLS AND TECHNIQUES, ADA

This report discusses software maintenance and proposes maintenance tools and techniques for the Ada* programming environment. Maintenance practices for several Air Force Command, Control, Communications, and Intelligence (C3I) software projects are reviewed. Three out of the four major problems identified during the project were attributed to the difficulty of comprehending software. Nine tools are proposed to help solve these and other problems, including a tool to help coordinate the programming process, a tool to aid in the collection and use of documentation, and an editor that is knowledgeable about what it is editing. The tools are based on related technologies that are also discussed in the report: artificial intelligence, automatic programming, intelligent use interfaces, formal verification, programming environments, and software metrics. Recommendations are made as to how the tools may be incorporated into the Ada Programming Support Environment (APSE). (*Ada is a trademark of the U.S. Department of Defense).

6176 VAN DERLINDEN, PETER; "WRITING DIAGNOSTIC SOFTWARE IN ADA," ACM ADA LETTERS. 4(2): Sep 1984. PP. 44-53.

Keywords: TESTING, VERIFICATION TOOLS AND TECHNIQUES, EXPERT SYSTEMS, ARCHITECTURE, ADA

This paper desribes the VERIFY 432 package, which was written in Ada*, and which evaluates the hardware status of an HIS 432 board system or an integrated MULTIBOX computer. Some observations are made on the possibility of building a knowledge base into this software, to upgrade to an expert system. A hardware diagnostic package would usually be written in a low-level language, perhaps even utilising special-purpose microcode functions. However, the authors' experience demonstrated that Ada is suitable for implementing this kind of testing suite, and has many features which especially facilitate more general systems programming. Some of the particular advantages of using Ada are pointed out, as well as areas in which the language could have provided more assistance than it did. (author) (*Ada is a trademark of the U.S. Department of

Defense).

6907 KEIRSEY, D.; MITCHELL, J.; BULLOCK, B.; NUSSMEIER, T.; TSENG, D.; AUTONOMOUS VEHICLE CONTROL USING ARRTIFICIAL INTELLIGENCE TECHNIQUES. Avail. from National Technical Information Service 5285 Port Royal Rd, Springfield, VA 22161. No. AD-P003 033.

Keywords: SYSTEM TESTING, ROBOTICS, ARTIFICIAL INTELLIGENCE

A review of early work on a project to develop autonomous vehicle control technology is presented. The primary goal of this effort is the development of a generic capability that can be specialized to a wide range of DoD applications. The emphasis in this project is development of the fundamental Artificial Intelligence based technology required by autonomous systems and the implementation of a testbed environment to evaluate and demonstrate the system capabilities. (author)

END
FILMED
FEB. 1988
DTIC